ПРИЛОЖЕНИЯ

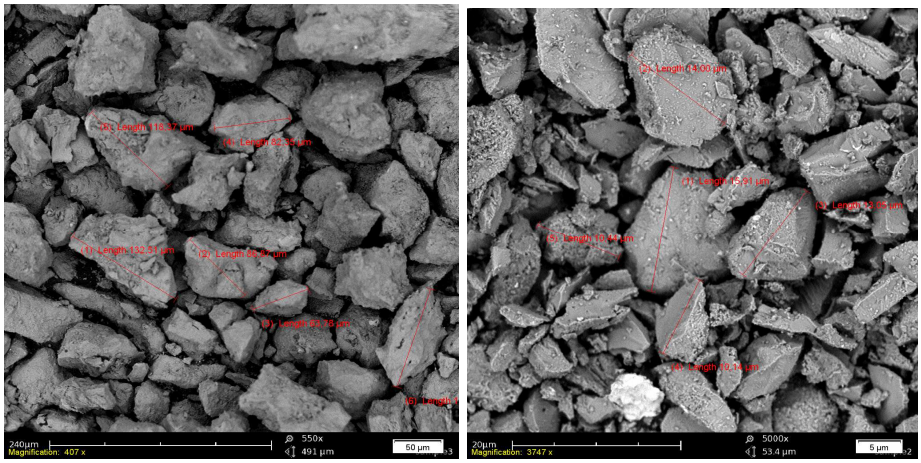
ПРИЛОЖЕНИЕ А

Результаты исследований гранулометрического состава продуктов струйного измельчения

		Размер класса								
№ пробы	материал	4-3,5	3-1,6	1,6-1,0	1-0,63	0,63-0,4	0,4-0,2	0,2-0,1	0,1-0,063	-0,006
1	2	3	4	5	6	7	8	9	10	11
0	этал шамот							5,1	11,5	83,4
%								0,77	0,92	2,502
1								0,65	3,6	95,75
%								0,098	0,288	2,8725
2								3,3	7,8	88,9
%								0,495	0,624	2,667
11	исх шамот	3,5	28,2	15,8	10,7	7,6	8,3	5,7	3,2	17
%		12,3	64,86	20,54	8,56	3,8	2,49	0,86	0,256	0,51
10	ОПК	0	15,2	24,9	20,2	14,5	14	5,6	2,3	3,3
%		0	34,96	32,37	16,16	7,25	4,2	0,84	0,184	0,099
12	шамот ОПК			0,6	3,1	4,4	19,4	16,9	9,1	46,5
%				0,78	2,48	2,2	5,82	2,54	0,728	1,395
13	шамот ОПК			1,2	2,5	3	11,1	12,2	8,9	61,1
%				1,56	2	1,5	3,33	1,83	0,712	1,833
49	шамот	0,9	3,5	8,9	13	12,5	37,8	12,2	3,7	7,5
%	ОПК	2,88	7	11,57	10,4	6,25	11,3	1,83	0,296	0,225
52	шамот		1,1	4	13	16,6	39,5	9,9	4,1	11,8
%	ОПК		2,2	5,2	10,4	8,3	11,9	1,49	0,328	0,354
22	шамот						1,1	4,6	11,9	82,4
%							0,33	0,69	0,952	2,472
21	шамот			2,8	5,7	9,1	36,2	25,9	10,3	10
%				3,64	4,56	4,55	10,9	3,89	0,824	0,3
25	шамот						0,4	1,7	4,9	93
%							0,12	0,255	0,392	2,79
1	исх шамот	2,7	20,4	25,9	16,9	8,7	13,4	5,6	2,1	4,3
%		8,64	40,8	33,67	13,52	4,35	4,02	0,84	0,168	0,129
30	песок пол								1,5	98,5
%	ОПК								0,12	2,955
27	песок пол							0,3	0,3	99,4
%								0,045	0,024	2,982
31	песок пол						0,3	0,9	2	96,8
%							0,09	0,14	0,16	2,904
29	песок пол						6,2	17,2	13,2	62,9
%							1,86	2,58	1,056	1,887
26	песок пол						10,7	23,7	14	51,6
%	ОПК						3,21	3,56	1,12	1,548
28	песок пол					0,3	29,7	46,4	15,6	17
%	ОПК					0,15	8,91	6,96	1,248	0,51

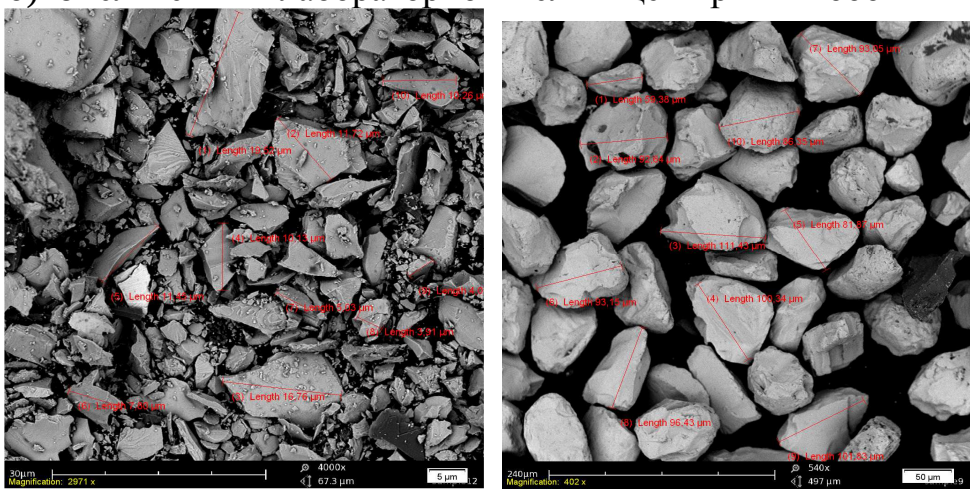
1	2	3	4	5	6	7	8	9	10	11
32	исх песок полигон				0,4	1,3	47,8	47,3	2,6	0,6
%					0,32	0,65	14,3	7,095	0,208	0,018
41	песок								0,1	99,9
%	циклон								0,008	2,997
47	песок		1,2	2,5	4	6	27,2	21	9,4	28,7
%	ОПК		2,4	3,25	3,2	3	8,16	3,15	0,752	0,861
2	песок исх				2,6	3,9	64	25,2	2,1	2,2
%	ВВ		0	0	2,08	1,95	19,2	3,78	0,168	0,066
39	цемент								0,47	99,53
%									0,0376	2,9859
38	цемент								0,25	99,75
%									0,02	2,9925
40	цемент							5,2	3,7	91,1
%	ОПК							0,78	0,296	2,733
37	цемент							3,6	7,5	86,9
%	исходн							0,54	0,6	2,607
61	цемент						0,8	2,8	8	88,4
%	исходн						0,24	0,42	0,64	2,652
54	песок ОПК					5,8	10,6	7,4	3,7	72,5
%						5,8	10,6	7,4	3,7	72,5
	Шлак отв исх		8,45	8,4	7,34	7,35	9,38	9,35	2,72	4,55
%			14,685	14,5985	12,756	12,77	16,3	16,2496	4,72715	7,90754
	шлак сталепл исх		4,92	5,52	4,23	4,2	7,4	12,9	6,14	14,22
%			8,2647	9,27264	7,1057	7,055	12,43	21,67	10,3141	23,8871
	ТГШ после шаров		0,1	0,27	0,23	0,29	1,11	21	16,3	65,33
%			0,0956	0,25805	0,2198	0,277	1,061	20,07	15,5787	62,4391
	шлак изм		0	0	0	0,1	0,6	0,75	0,84	25
%	3000 об					0,366	2,199	2,748	3,07805	91,6086
%	дом отв	0	7,8	8,4	10,8	9,6	24,1	15,3	9,2	3,1
%	сталепл	0	8	5,6	6	3,8	12,2	14,5	7,4	3
%	домен	0	1,6	3	5,2	3,6	10,3	17,4	23,5	8,7
%	техн углер	1,7	7,8	9,4	11,3	9,1	24,9	17,4	4,8	13,6
%	техн углер	0,5	7,9	9,6	11,6	8,4	28	19,2	6,7	8,1

Фото проб материалов на электронном микроскопе



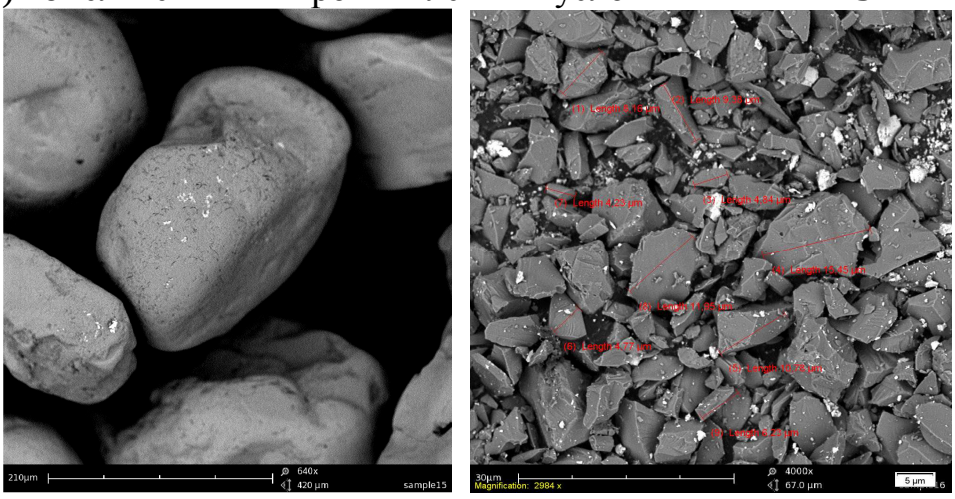
а) б)

Шлак: а) –исходный,
б)измельченный лабораторной мельницей при n=2000мин$^{-1}$



а) б)

Циркон: а) исходный,
б) измельченный в промышленных условиях ВГМК -45 мкм



а) б)

Кварцевый песок (Бразилия): а) – исходный,
б) измельченный лабораторной мельницей при n=2000мин$^{-1}$

ПРИЛОЖЕНИЕ Б

ПРОГРАММА ИМИТАЦИОННОЙ МОДЕЛИ ЗАМКНУТОГО ЦИКЛА ТОНКОГО ИЗМЕЛЬЧЕНИЯ

Имитационной моделью называют абстрактную динамическую модель, реализованную, как правило, на ЭВМ, и воспроизводящую в рамках установленных ограничений поведение оригинала в хронологическом порядке.

Имитационное моделирование является экспериментальной и прикладной методологией, имеющей целью:

1) описать поведение систем;

2) построить теории и гипотезы, которые могут объяснить наблюдаемое поведение;

3) использовать эти теории для предсказания будущего поведения системы, т.е. тех воздействий, которые могут быть вызваны изменениями в системе или изменениями способов ее функционирования

В приложении представлена разработка программного обеспечения (ПО) для имитационного моделирования кинетики движения материала при непрерывном измельчении в замкнутой технологической системе, включающей бункер-питатель, мельницу и классификатор.

Разрабатываемое ПО обеспечивает выполнение эксперимента над разработанной моделью путем варьирования глобальных параметров модели , получает зависимости этих параметров от других параметры во времени и выполняет следующие дополнительные функции:

- отображает основные элементы технологической схемы;
- отображает характеристики гранулометрического состава на каждом элементе технологической схемы;
- показывает величины потоков в ветвях технологической схемы;

– отображает значение рабочего заполнения мельницы в течении всего периода моделирования;
– отображает в виде временной зависимости значения расхода исходного и готового материала;
– позволяет изменять величину подачи готового материла в зависимости от соотношения крупности отдельных фракций до и после измельчения.

Внедрение предлагаемой имитационной модели может позволить уменьшить затраты на определения оптимальных режимов работы измельчительного оборудования. Программное обеспечение приложения реализовано с использованием языка высокого уровня JAVA.

Модель представляет собой набор следующих файлов:

– **.bat** файл, используемый для запуска автономного приложения;
– **ф**айл исполняющего модуля (model**.jar**);
– Файлы библиотек сторонних разработчиков ПО (xjanylogic6engine.jar и xjanylogic6engine_ru.jar);
– Файл библиотеки EnterpriseLibrary(EnterpriseLibrary.jar);
– HTML файл, используемый для запуска Java апплета (model2_4.html)

Описание программного обеспечения

Первым окном программы является окно презентации эксперимента , которое автоматически отображается перед запуском презентации модели (см. рис. Б1). Вначале на экране появляется презентация запущенного эксперимента. В этот момент модель еще не создана и создан только сам эксперимент. В этом окне инициализируются переменные модели, которые связываются с параметрами модели. Запуск модели осуществляется кнопкой Пуск.

На рис. Б.1 показано окно презентации эксперимента, в котором можно установить следующие варьируемые параметры запускаемой модели:

– коэффициенты, определяющие грансостав поступающего на измельчение материала;

– объем мельницы

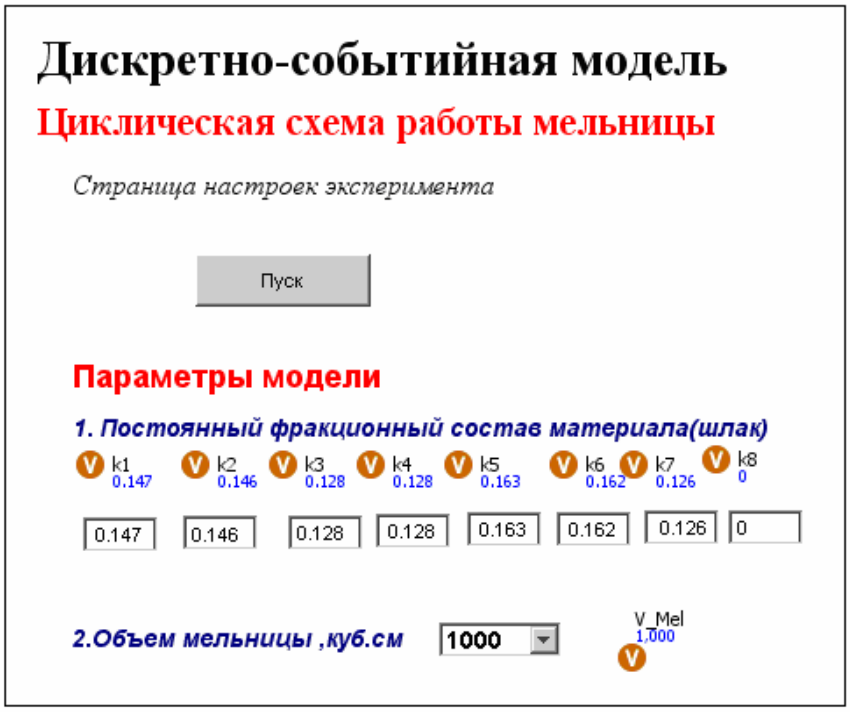


Рисунок Б.1 – Первичное окно презентации эксперимента

Все переменные, параметры и пользовательские функции, показаные в окне презентации не отображаются, но могут быть просмотрены в среде AnyLogic. Представленные пользовательские функции являются вспомогательными и предназначены в основном для выода промежуточных расчетных значений.

Окно презентации модели (см.рис. Б.3) состоит из двух панелей – панели управления, панели состояния и области отображения модели мельницы. Панель управления находится в верхней части окна презентации, а панель состояния - в нижней. Панель управления содержит секции с различными командами, позволяющими управлять выполнением модели, представлением презентации модели, осуществлять быструю навигацию по объектам модели и т.д. Область отображения модели содержит графическое представление элементов технологической схемы.

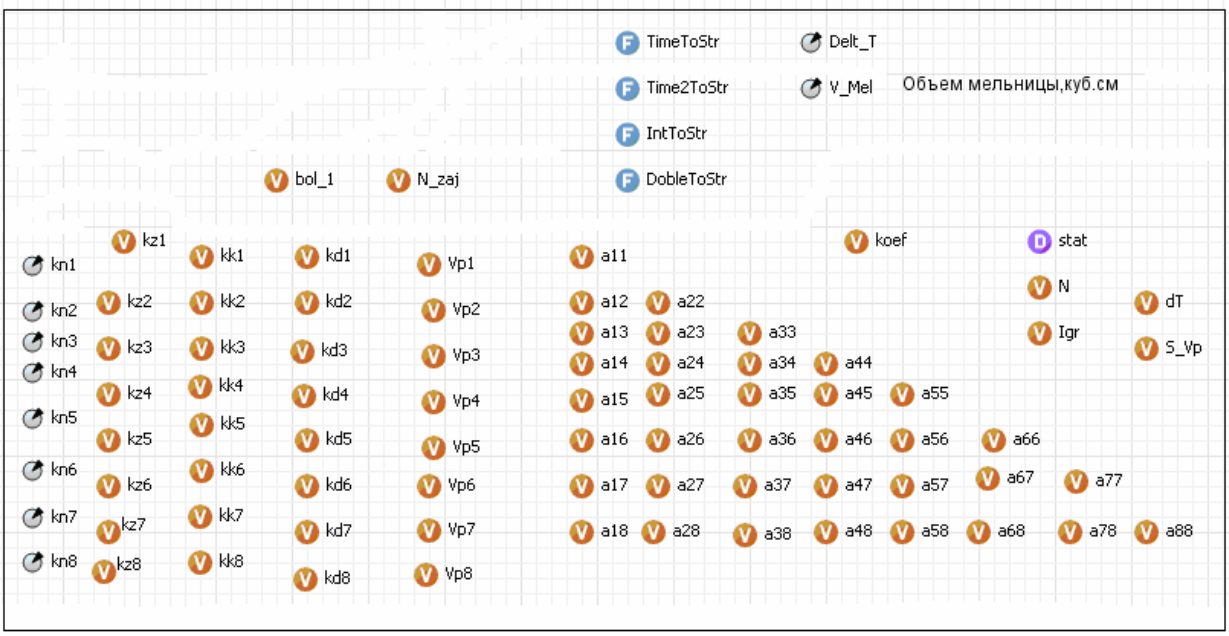


Рисунок Б.2 – Переменные и параметры модели

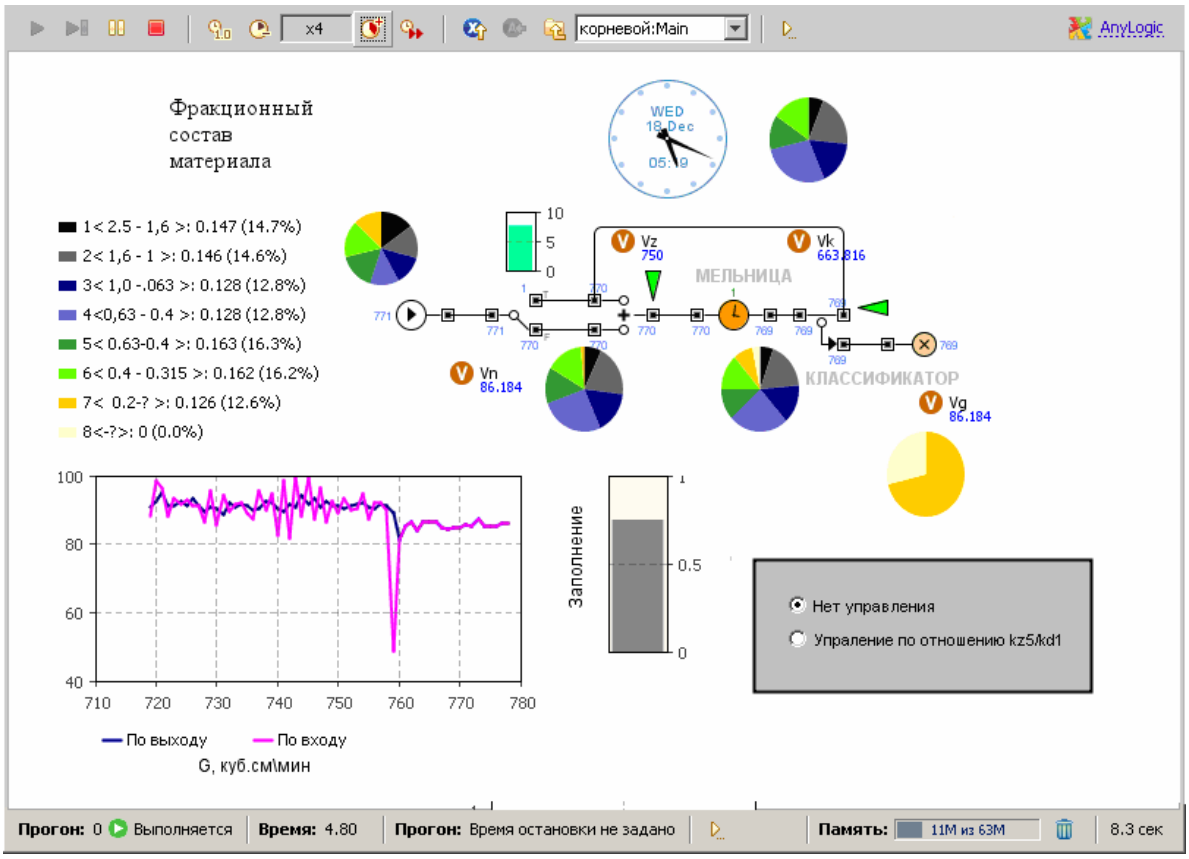
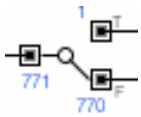
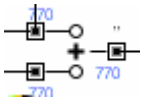


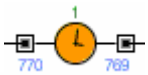
Рисунок Б.3 – Окно презентации модели

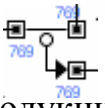
На рисунке Б.3 показана презентация модели, в которой применяются следующие графические обозначения:

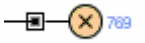
– Питатель подачи исходного материала – 

– Переключатель потока в момент начальной загрузки – 

– Смеситель потоков материала – 

– Струйная мельница – 

– Классификатор – 

– Бункер готовой продукции – 

Таким образом, в приложении показана разработанная структурная схема имитационной модели кинетики измельчения в замкнутого цикле, основанной на дискретно-событийном подходе.

В инструментальной среде AnyLogic разработано программное обеспечение, позволяющее выполнить компьютерные эксперименты и провести оптимизацию параметров относительно заданного критерия.

Практическая значимость создания данной модели состоит в том, что впервые показана возможность моделирования кинетики изменения гранулометрического состава материала в технологических схемах измельчения при помощи среды моделирования AnyLogic.

Листинг программного обеспечения

```
package model2;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import com.xj.anylogic.libraries.enterprise.*;
import java.awt.geom.Arc2D;
public class Main extends ActiveObject
{
  // Параметры
  public
int  V_Mel;
  /**
   * Возвращает значение по умолчанию параметра <code>V_Mel</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  public
int  _V_Mel_DefaultValue_xjal() {
    return 0;
  }
  public void set_V_Mel(
```

```
int  V_Mel ) {
    if (V_Mel == this.V_Mel) {
      return;
    }
    this.V_Mel = V_Mel;
    onChange_V_Mel();
    onChange();
  }
  void onChange_V_Mel() {
  }
  public
double  kn1;
  /**
   * Возвращает значение по умолчанию параметра <code>kn1</code>.
  // События
  public EventTimeout _stat_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _chart_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _plot_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _chart2_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _chart6_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _plot1_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _chart1_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _chart3_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _chart4_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _chart5_autoUpdateEvent_xjal = new EventTimeout(this);
  @Override
  public String getNameOf( EventTimeout _e ) {
    if ( _e == _stat_autoUpdateEvent_xjal ) return "stat auto update event";
    if ( _e == _chart_autoUpdateEvent_xjal ) return "chart auto update event";
    if ( _e == _plot_autoUpdateEvent_xjal ) return "plot auto update event";
    if ( _e == _chart2_autoUpdateEvent_xjal ) return "chart2 auto update event";
    if ( _e == _chart6_autoUpdateEvent_xjal ) return "chart6 auto update event";
    if ( _e == _plot1_autoUpdateEvent_xjal ) return "plot1 auto update event";
    if ( _e == _chart1_autoUpdateEvent_xjal ) return "chart1 auto update event";
    if ( _e == _chart3_autoUpdateEvent_xjal ) return "chart3 auto update event";
    if ( _e == _chart4_autoUpdateEvent_xjal ) return "chart4 auto update event";
    if ( _e == _chart5_autoUpdateEvent_xjal ) return "chart5 auto update event";
     return super.getNameOf( _e );
  }
  @Override
  public int getModeOf( EventTimeout _e ) {
    if ( _e == _stat_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _plot_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart2_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart6_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _plot1_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart1_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart3_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart4_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart5_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    return super.getModeOf( _e );
  }
  @Override
  public double getFirstOccurrenceTime( EventTimeout _e ) {
    if (
        _e == _stat_autoUpdateEvent_xjal
        || _e == _chart_autoUpdateEvent_xjal
        || _e == _plot_autoUpdateEvent_xjal
        || _e == _chart2_autoUpdateEvent_xjal
        || _e == _chart6_autoUpdateEvent_xjal
        || _e == _plot1_autoUpdateEvent_xjal
        || _e == _chart1_autoUpdateEvent_xjal
        || _e == _chart3_autoUpdateEvent_xjal
```

```
        || _e == _chart4_autoUpdateEvent_xjal
        || _e == _chart5_autoUpdateEvent_xjal
      ) return getEngine().getStartTime();
    return super.getFirstOccurrenceTime( _e );
  }
  @Override
  public double evaluateTimeoutOf( EventTimeout _e ) {
    if ( _e == _stat_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _chart_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _plot_autoUpdateEvent_xjal ) return 1;
    if ( _e == _chart2_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _chart6_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _plot1_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _chart1_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _chart3_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _chart4_autoUpdateEvent_xjal ) return 1 ;
    if ( _e == _chart5_autoUpdateEvent_xjal ) return 1 ;
    return super.evaluateTimeoutOf( _e );
  }
  @Override
  // Вложенные объекты
  public Delay<
Zajvka > delay;
  public Source<
Zajvka > source;
  public Split<
Zajvka,Zajvka > split;
  public Sink<
Zajvka > sink3;
  public Combine<
Zajvka, Zajvka,Zajvka > combine;
  public SelectOutput<
Zajvka > selectOutput1;
  public Clock clock;
  public Batch<
Entity, Entity > batch;
  public Unbatch<
Entity, Entity > unbatch;
  public String getNameOf( ActiveObject ao ) {
    if ( ao == delay ) return "delay";
    if ( ao == source ) return "source";
    if ( ao == split ) return "split";
    if ( ao == sink3 ) return "sink3";
    if ( ao == combine ) return "combine";
    if ( ao == selectOutput1 ) return "selectOutput1";
    if ( ao == clock ) return "clock";
    if ( ao == batch ) return "batch";
    if ( ao == unbatch ) return "unbatch";
    return null;
  }
  public String getNameOf( ActiveObjectCollection<?> aolist ) {
    return null;
  }
  /**
   * Создает экземпляр вложенного объекта<br>
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private Delay<
Zajvka > instantiate_delay_xjal() {
    Delay<
Zajvka > object = new Delay<
Zajvka >( getEngine(), this, null ) {
      @Override
      public double delayTime( Zajvka entity ) {
```

```
        return _delay_delayTime_xjal( entity );
      }
      @Override
      public void onEnter( Zajvka entity, double delayTime ) {
        _delay_onEnter_xjal( entity, delayTime );
      }
      /**
       * Это число используется при сохранении состояния модели
       */
      private static final long serialVersionUID = 1379151964237L;
    };
        return object;
  }
  /**
   * Инициализация параметров экземпляра вложенного объекта<br>
   * Пользователь не должен вызывать этот метод
   */
  private void setupParameters_delay_xjal(Delay<
Zajvka > object ) {
    object.delayTimeDefinedByPath = ob-
ject._delayTimeDefinedByPath_DefaultValue_xjal();
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity = object._maximumCapacity_DefaultValue_xjal();
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType =
Animator.BAG ;
    object.animationForward = object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }
  /**
   * Инициализация экземпляра вложенного объекта<br>
   * Пользователь не должен вызывать этот метод
   */
  private void setupParameters_batch_xjal(Batch<
Entity, Entity > object ) {
    object.batchSize = object._batchSize_DefaultValue_xjal();
    object.permanent = object._permanent_DefaultValue_xjal();
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation = object._enableRotation_DefaultValue_xjal();
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward = object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }
  /**
  public void _delay_onEnter_xjal( Zajvka entity, double delayTime ) {
   // расчет новых обьемов фракций
double vvv=entity.V_Zaj;
double kl1,kl2,kl3,kl4,kl5,kl6,kl7,kl8;
kl1=entity.kf1;
kl2=entity.kf2;
kl3=entity.kf3;
kl4=entity.kf4;
kl5=entity.kf5;
kl6=entity.kf6;
kl7=entity.kf7;
kl8=entity.kf8;
Vp1=vvv*kz1*a11;
Vp2=vvv*(100*a12*kl1+100*a22*kl2)/100;
Vp3=vvv*(100*a13*kl1+100*a23*kl2+100*a33*kl3)/100;
Vp4=vvv*(100*a14*kl1+100*a24*kl2+100*a34*kl3+100*a44*kl4)/100;
Vp5=vvv*(100*a15*kl1+100*a25*kl2+100*a35*kl3+100*a45*kl4 + 100*a55*kl5)/100;
Vp6=vvv*(100*a16*kl1+100*a26*kl2+100*a36*kl3+100*a46*kl4 +
100*a56*kl5+100*a66*kl6)/100;
```

```
Vp7=vvv*(100*a17*kl1+100*a27*kl2+100*a37*kl3+100*a47*kl4 +
100*a57*kl5+100*a67*kl6+100*a77*kl7)/100;
Vp8=vvv*(100*a18*kl1+100*a28*kl2+100*a38*kl3+100*a48*kl4 +
100*a58*kl5+100*a68*kl6+100*a78*kl7+100*a88*kl8)/100;
double sss=Vp1+Vp2+Vp3+Vp4+Vp5+Vp6+Vp7+Vp8;
S_Vp=sss;
kd1=Vp1/sss;kd2=Vp2/sss;
kd3=Vp3/sss;kd4=Vp4/sss;
kd5=Vp5/sss;kd6=Vp6/sss;
kd7=Vp7/sss;kd8=Vp8/sss;
entity.V_Zaj=sss;
entity.kf1=kd1;
entity.kf2=kd2;
entity.kf3=kd3;
entity.kf4=kd4;
entity.kf5=kd5;
entity.kf6=kd6;
entity.kf7=kd7;
entity.kf8=kd8;
;
}
  public Entity _source_newEntity_xjal(  ) {
    return
new  Zajvka()
;
  }
  public void _source_onExit_xjal( Zajvka entity ) {
    if(N_zaj==0)
{
Vk=0;
//Vk=0.75*V_Mel;
entity.V_Zaj=Vk; N_zaj=1; bol_1=true;
entity.kf1=0;entity.kf2=0;entity.kf3=0;
entity.kf4=0;entity.kf5=0;entity.kf6=0;
entity.kf7=0;entity.kf8=0;
}
else
{
bol_1=false;
entity.V_Zaj=Vn;//kn3=triangular(0.9*kn3,1.1*kn3,kn3);
entity.kf1=kn1;entity.kf2=kn2;entity.kf3=kn3;
entity.kf4=kn4;entity.kf5=kn5;entity.kf6=kn6;
entity.kf7=kn7;entity.kf8=kn8;
}
;
  }
  public Entity _split_newEntity_xjal( Zajvka original, int index ) {
    return
new Zajvka()
;
  }
  public void _split_onExit_xjal( Zajvka entity ) {
//--------------------
// Выход оригинала
//--------------------
Vk=Vz*(kd1+kd2+kd3+kd4+kd5+kd6);
kk1=kd1*Vz/Vk;
kk2=kd2*Vz/Vk;
kk3=kd3*Vz/Vk;
kk4=kd4*Vz/Vk;
kk5=kd5*Vz/Vk;;
kk6=kd6*Vz/Vk;
kk7=0;
kk8=0;
```

```
Vg=Vz*(kd7+kd8);
entity.V_Zaj=Vk;
entity.kf1=kk1;
entity.kf2=kk2;
entity.kf3=kk3;
entity.kf4=kd4;
entity.kf5=kd5;
entity.kf6=kk6;
entity.kf7=kk7;
entity.kf8=kk8;
double ttt;
//if((N%2)==0)
{
ttt=0.75*V_Mel-Vk;
//if(koef==0)ttt=ttt;
if(koef==1)ttt=ttt*kz5/(kz1);
Vn=ttt;
Vn=triangular((1-neravn_pit)*ttt,(1+neravn_pit)*ttt,ttt);
}
N=N+1;
source.inject(1);
 ;
  }
  public Entity _combine_newEntity_xjal( Zajvka entity1, Zajvka entity2 ) {
    return
new Zajvka()
;
  }
  public void _combine_onExit_xjal( Zajvka entity, Zajvka entity1, Zajvka en-
tity2 ) {
//------------------------------
// Случайная концентрация
//------------------------------
kper1=triangular(0.8*kn1,1.2*kn1,kn1);
kper2=triangular(0.8*kn2,1.2*kn2,kn2);
kper3=triangular(0.8*kn3,1.2*kn3,kn3);
kper4=triangular(0.8*kn4,1.2*kn4,kn4);
kper5=triangular(0.8*kn5,1.2*kn5,kn5);
kper6=triangular(0.8*kn6,1.2*kn6,kn6);
kper7=triangular(0.8*kn7,1.2*kn7,kn7);
kper8=triangular(0.8*kn8,1.2*kn8,kn8);
 sum_kp=kper1+kper2+kper3+kper4+kper5+kper6+kper7+kper8;
//==============================
Vz=Vn+Vk;
kz1=(kper1*Vn+kk1*Vk)/Vz;entity.kf1=kz1;
kz2=(kper2*Vn+kk2*Vk)/Vz;entity.kf2=kz2;
kz3=(kper3*Vn+kk3*Vk)/Vz;entity.kf3=kz3;
kz4=(kper4*Vn+kk4*Vk)/Vz;entity.kf4=kz4;
kz5=(kper5*Vn+kk5*Vk)/Vz;entity.kf5=kz5;
kz6=(kper6*Vn+kk6*Vk)/Vz;entity.kf6=kz6;
kz7=(kper7*Vn+kk7*Vk)/Vz;entity.kf7=kz7;
kz8=(kper8*Vn+kk8*Vk)/Vz;entity.kf8=kz8;
entity.V_Zaj=Vz;
;
  }
  public boolean _selectOutput1_condition_xjal( Zajvka entity ) {
    return
bol_1 ;
  }
  public double _selectOutput1_probability_xjal( Zajvka entity ) {
    return
1;
  }
  // Функции
```

```
String
 TimeToStr(  ) {
String b;
double a=time();
b=Double.toString(a);
return b;
  }
String
 Time2ToStr(  ) {
String b,b1,b2,b3;
double s=time();
int h=(int)s / 3600;
int k=(int)s-h*3600;
int m=k/60;
int sec=k-m*60;
b1=Integer.toString(h);
b2=Integer.toString(m);if(b2.length()==1)b2="0"+b2;
b3=Integer.toString(sec);if(b3.length()==1)b3="0"+b3;
return (b1+" : "+b2+" : "+b3);
   }
String
 IntToStr( int x ) {
String b;
b=Integer.toString(x);
return b;
   }
String
 DobleToStr( double x ) {
String b;
b=Double.toString(x);
return b;
 }
  // Объекты статистики
  public DataSet _plot_expression0_dataSet_xjal = new DataSet( 60 ) {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
Vg ;
      add( time(), _a );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели
     */
    private static final long serialVersionUID = 0L;
  };
  public DataSet _plot_expression1_dataSet_xjal = new DataSet( 60 ) {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
Vn ;
      add( time(), _a );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели
     */
    private static final long serialVersionUID = 0L;
  };
  public DataSet _plot1_expression0_dataSet_xjal = new DataSet( 100 ) {
```

```
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
kz5/kz1
;
      add( time(), _a );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели
     */
    private static final long serialVersionUID = 0L;
  };
  public StatisticsDiscrete stat = new StatisticsDiscrete() {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      add( _stat_Value() );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели
     */
    private static final long serialVersionUID = 1382711607641L;
  };

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _stat_Value() {
    return
Vg
;
  }
  /**
   * Идентификатор группы presentation верхнего уровня
   */
  static final int _presentation = 0;
  /**
   * Идентификатор группы icon верхнего уровня
   */
  static final int _icon = -1;

  static final double[] _polyline_pointsDX_xjal() {
    return new double[] { 0, 10, 5, 0,  };
  }
  static final double[] _polyline_pointsDY_xjal() {
    return new double[] { 0, 0, 20, 0,  };
  }
  static final double[] _polyline1_pointsDX_xjal() {
    return new double[] { 0, 0, -20, 0,  };
  }
  static final double[] _polyline1_pointsDY_xjal() {
    return new double[] { 0, 10, 5, 0,  };
  }
  @Override
  public void executeShapeControlAction( int _shape, int index ) {
    switch( _shape ) {
      case _button: {
source.inject(1);
;}
```

```
        break;
      default:
        super.executeShapeControlAction( _shape, index );
        break;
    }  }
  @Override
  public void executeShapeControlAction( int _shape, int index, int value ) {
    switch( _shape ) {
      case _radio:
        koef = value;
        break;
      default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }  }
  @Override
  public void executeShapeControlAction( int _shape, int index, double value ) {
    switch( _shape ) {
      case _slider:
        dT = (int) value;
        break;
      case _slider1:
        neravn_pit = value;
        break;
      default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }  }
   @Override
  public double getShapeControlMinimum( int _shape, int index ) {
    switch( _shape ) {
      case _slider: return
1 ;
      case _slider1: return
0 ;
      default: return super.getShapeControlMinimum( _shape, index );
    }  }
  @Override
  public double getShapeControlMaximum( int _shape, int index ) {
    switch( _shape ) {
      case _slider: return
10 ;
      case _slider1: return
0.5
;
      default: return super.getShapeControlMaximum( _shape, index );
    }  }
  @Override
  public int getShapeControlDefaultValueInt( int _shape, int index ) {
    switch(_shape) {
      case _radio: return
koef
;
      default: return super.getShapeControlDefaultValueInt( _shape, index );
    }  }
  @Override
  public double getShapeControlDefaultValueDouble( int _shape, int index ) {
    switch(_shape) {
      case _slider: return
dT
;
      case _slider1: return
neravn_pit
;
```

```
      default: return super.getShapeControlDefaultValueDouble( _shape, index );
    }  }
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem0Value() {
    return
kn1
;  }
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem1Value() {
    return kn2 ;
  }
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem2Value() {
    return kn3 ;
  }
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem3Value() {
    return kn4 ;
  }
    /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem4Value() {
    return kn5 ;
  }
    /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem5Value() {
    return kn6 ;
  }
    /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem6Value() {
    return kn7 ;
  }
    /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart_DataItem7Value() {
    return kn8 ;
  }
    /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart2_DataItem0Value() {
    return Vz/V_Mel ;  }
    /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  private double _chart6_DataItem0Value() {
    return Vk/Vn ;
  }
   * <i>Пользователь не должен вызывать этот метод</i>
   */
```

```
  private void _slider1_SetDynamicParams_xjal( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _slider1 ), getShapeControlMaximum(
_slider1 ) );
  }
    ShapeSlider slider1;
  PieChart chart;

  // Статическая инициализация элементов, у которых разрешено программное управ-
ление
  {
    button = new ShapeButton(
                 Main.this, true, 250, 40,
                 110, 30,
            controlDefault, controlDefault,
            _button_Font,
                 "Включить"
    ) {
      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
        _button_SetDynamicParams_xjal( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
      @Override
      public void action(){
        executeShapeControlAction( _button, 0 );
      }
      /**
       * Это число используется при сохранении состояния модели
       */
      private static final long serialVersionUID = 1382082191657L;
    };
    slider = new ShapeSlider(
                Main.this, true,640, 140,
                  100, 30,
            transparent,
            false, getShapeControlMinimum( _slider ), getShapeControlMaximum(
_slider ), ShapeControl.TYPE_INT
    ) {
      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
        _slider_SetDynamicParams_xjal( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
      @Override
      public void action(){
        executeShapeControlAction( _slider, 0, value );
      }
      @Override
      public void setValueToDefault() {
           setValue( limit( getMin(), getShapeControlDefaultValueDouble(
_slider, 0 ), getMax() ) );
      }
      /**
       * Это число используется при сохранении состояния модели
       */
      private static final long serialVersionUID = 1382697501704L;
    };
    radio = new ShapeRadioButtonGroup(
                Main.this, true,750, 210,
                  150, 100,
            transparent, controlDefault,
```

```
            _radio_Font, true,
            new String[]{"-", "kz5/kd1", }
    ) {
      @Override
      public void action(){
        executeShapeControlAction( _radio, 0, value );
      }
      @Override
      public void setValueToDefault() {
            setValue( getShapeControlDefaultValueInt( _radio, 0 ) );
      }
      /**
       * Это число используется при сохранении состояния модели
       */
      private static final long serialVersionUID = 1385107631954L;
    };
    slider1 = new ShapeSlider(
                Main.this, true,490, 340,
                  100, 30,
            transparent,
            false, getShapeControlMinimum( _slider1 ), getShapeControlMaximum(
_slider1 ), ShapeControl.TYPE_DOUBLE
    ) {
      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
        _slider1_SetDynamicParams_xjal( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
      @Override
      public void action(){
        executeShapeControlAction( _slider1, 0, value );
      }
      @Override
      public void setValueToDefault() {
            setValue( limit( getMin(), getShapeControlDefaultValueDouble(
_slider1, 0 ), getMax() ) );
      }
      /**
       * Это число используется при сохранении состояния модели
       */
      private static final long serialVersionUID = 1385108109641L;
    };
    text1 = new ShapeText(
        true,40, -710, 0.0,
        black,"ПАРАМЕТРЫ",
        _text1_Font, ALIGNMENT_LEFT    );
    text11 = new ShapeText(
        true,410, -720, 0.0,
        black,"ПЕРЕМЕННЫЕ",
        _text11_Font, ALIGNMENT_LEFT    );
    text13 = new ShapeText(
        true,20, -660, 0.0,
        black,"Объем мельницы",
        _text13_Font, ALIGNMENT_LEFT    );
    text14 = new ShapeText(
        true,20, -640, 0.0,
        black,"Начальная подача,El_Ob/c",
        _text14_Font, ALIGNMENT_LEFT    );
    text15 = new ShapeText(
        true,20, -620, 0.0,
        black,"Плотность материала",
        _text15_Font, ALIGNMENT_LEFT    );
    text10 = new ShapeText(
```

```
        true,280, 20, 0.0,
        silver,"МЕЛЬНИЦА",
        _text10_Font, ALIGNMENT_LEFT    );
    text16 = new ShapeText(
        true,520, 10, 0.0,
        silver,"КЛАССИФИКАТОР",
        _text16_Font, ALIGNMENT_LEFT    );
    text6 = new ShapeText(
        true,600, -350, 0.0,
        black,"Объем мельницы,куб.см",
        _text6_Font, ALIGNMENT_LEFT    );
    text = new ShapeText(
        true,110, 30, 0.0,
        black,"Фракционный \r\nсостав\r\nматериала",
        _text_Font, ALIGNMENT_LEFT    );
    polyline = new ShapePolyLine(
            true,435, 150,
            black, lime,
            4, _polyline_pointsDX_xjal(), _polyline_pointsDY_xjal(),
            false, 1, LINE_STYLE_SOLID
    );
    polyline1 = new ShapePolyLine(
            true,600, 170,
            black, lime,
            4, _polyline1_pointsDX_xjal(), _polyline1_pointsDY_xjal(),
            false, 1, LINE_STYLE_SOLID
    );
    text2 = new ShapeText(
        true,380, 380, 4.71238898038469,
        black,"Заполнение",
        _text2_Font, ALIGNMENT_LEFT
    );
    text3 = new ShapeText(
        true,130, 480, 0.0,
        black,"G, куб.см\\мин",
        _text3_Font, ALIGNMENT_LEFT
    );
  }
  ShapeGroup presentation;
  ShapeGroup icon;
  @Override
  public Object getPersistentShape( int _shape ) {
  static final int[] _connector_pointsX = {
    300,  330 };
  static final int[] _connector_pointsY = {
    180,180 };
  static final int[] _connector1_pointsX = {    360,  400 };
  static final int[] _connector1_pointsY = {    170,170 };
  static final int[] _connector2_pointsX = {    360,  400 };
  static final int[] _connector2_pointsY = {    190,190 };
  static final int[] _connector5_pointsX = {    570,  600 };
  static final int[] _connector5_pointsY = {    200,200 };
  static final int[] _connector6_pointsX = {    520,  540 };
  static final int[] _connector6_pointsY = {    180,180 };
  static final int[] _connector3_pointsX = {    440,  470 };
  static final int[] _connector3_pointsY = {    180,180 };
  static final int[] _connector4_pointsX = {    570,  570, 400, 400 };
  static final int[] _connector4_pointsY = {    180,120, 120, 170 };
  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean _publicOnly
) {
    if (!_publicOnly) {
      drawParameter( _panel, _g, 540, -340, 10, 0, "V_Mel", V_Mel, false, false
);
```

```
    }
    if (!_publicOnly) {
      drawParameter( _panel, _g, 540, -370, 10, 0, "Delt_T", Delt_T, false,
false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 480, 130, 10, 0, "S_Vp", S_Vp, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 680, -230, 10, 0, "koef", koef, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 500, 380, 10, 0, "neravn_pit", neravn_pit,
false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 270, -280, 10, 0, "N_zaj", N_zaj, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 190, -280, 10, 0, "bol_1", bol_1, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 310, 220, 10, 0, "Vn", Vn, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 540, 130, 10, 0, "Vk", Vk, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 630, 240, 10, 0, "Vg", Vg, false );
    }
if (!_publicOnly) {
      drawDataset( _panel, _g, 820, 60, 10, 0, "stat", stat );
    }
    // Embedded object "delay"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 420 , 130 , 70, 20, null,
this.delay );
    }
    // Embedded object "source"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 200 , 130 , 60, 20, null,
this.source );
    }
    // Embedded object "split"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 490 , 130 , 52, 27, null,
this.split );
    }
    // Embedded object "sink3"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 550 , 150 , 56, 30, null,
this.sink3 );
    }
    // Embedded object "combine"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 340 , 130 , 58, 17, null,
this.combine );
    }
    // Embedded object "selectOutput1"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 280 , 130 , 29, 17, null,
this.selectOutput1 );
    }
    // Embedded object "clock"
```

```
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 450 , 60 , -19, -51, null,
this.clock );
    }
    // Embedded object "batch"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, -140 , 190 , 58, 23, "batch",
this.batch );
    }
    // Embedded object "unbatch"
    if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, -150 , 260 , 50, 30, "unbatch",
this.unbatch );
    }
    if (!_publicOnly) {
      drawConnector( _panel, _g, _connector_pointsX, _connector_pointsY, false
);
      }
    if (!_publicOnly) {
      drawConnector( _panel, _g, _connector1_pointsX, _connector1_pointsY, false
);
  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int click-
Count, boolean publicOnly ) {
    if( !publicOnly && modelElementContains(x, y, 540, -340) ) {
      panel.addInspect( 540, -340, this, "V_Mel" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 540, -370) ) {
      panel.addInspect( 540, -370, this, "Delt_T" );
      return true;
      return true;
    }
    return false;
  }
  /**
   * Конструктор
   */
  public Main( Engine engine, ActiveObject owner, ActiveObjectCollection<? ex-
tends Main> collection ) {
    super( engine, owner, collection );
  }
   @Override
  public void create() {
    // Создание экземпляров вложенных объектов
    delay = instantiate_delay_xjal();
    source = instantiate_source_xjal();
    split = instantiate_split_xjal();
    sink3 = instantiate_sink3_xjal();
    combine = instantiate_combine_xjal();
    selectOutput1 = instantiate_selectOutput1_xjal();
    clock = instantiate_clock_xjal();
    batch = instantiate_batch_xjal();
    unbatch = instantiate_unbatch_xjal();
    // Присвоение начальных значений простым переменным
    N_zaj =
0 ;
    bol_1 = true
;
    Vn = 0.75*V_Mel;     Vk = 0 ;    Vg = 0;    Igr = 5 ;
    kk1 = 0 ;    kk2 = 0 ;    kk3 = 0;    kk4 = 0 ;    kk5 = 0 ;
    kk6 = 0 ;    kz1 = 0;    kz2 = 0 ;    kz3 = 0 ;    kz4 = 0 ;
    kz5 = 0 ;    kz6 = 0 ;    Vp1 = 0 ;    Vp2 = 0;    Vp3 = 0;
    Vp4 = 0 ;    Vp5 =0;    Vp6 = 0;    a11 = 0.764 ;    a12 = 0.197;
```

```
    a13 = 0.0181 ;    a14 = 0.01; a15 = 0.0042 ;    a16 = 0.005;    N = 0;
    a22 = 0.8448 ;    a23 = 0.117 ;    a24 = 0.023 ;    a25 = 0.007 ;
    a26 = 0.007 ;    a33 = 0.7595 ;    a34 = 0.1623;    a35 = 0.0238;
    a36 = 0.024 ;    a44 = 0.822 ;    a45 = 0.105;    a46 = 0.0416 ;
    a55 = 0.6152 ;    a56 = 0.2345 ;    a66 = 0.573;    kd1 = 0 ;    kd2 = 0 ;
    kd3 = 0 ;    kd4 = 0 ;    kd5 = 0;    kd6 = 0 ;    dT = 1;    kz7 = 0 ;
    kk7 = 0 ;    kd7 = 0 ;    Vp7 = 0;    a17 = 0.0017;    a27 = 0.001 ;
    a37 = 0.025 ;    a47 = 0.0214;    a57 = 0.1003 ;  a67 = 0.327 ;  a77 = 0.5;
    kz8 = 0 ;    kk8 = 0;    kd8 = 0 ;    Vp8 = 0 ;  a18 = 0.00 ;  a28 =0.0002;
    a38 = 0.0054;    a48 =0.01; a58 = 0.05 ;    a68 =0.1; a78 =0.5;  a88 = 1.0 ;
    kper1 = 0 ;    kper2 = 0 ;    kper3 =0 ;    kper4 = 0 ;    kper5 =0;
    kper6 = 0 ;    kper7 =0 ;    kper8 =0 ;    sum_kp = 0;
  }
  private void readObject(java.io.ObjectInputStream _stream) throws
java.io.IOException, ClassNotFoundException {
    _stream.defaultReadObject();
    delay.restoreOwner( this );
    source.restoreOwner( this );
    split.restoreOwner( this );
    sink3.restoreOwner( this );
    combine.restoreOwner( this );
    selectOutput1.restoreOwner( this );
    clock.restoreOwner( this );
    batch.restoreOwner( this );
    unbatch.restoreOwner( this );
    _stat_autoUpdateEvent_xjal.restoreOwner( this );
    _chart_autoUpdateEvent_xjal.restoreOwner( this );
    _plot_autoUpdateEvent_xjal.restoreOwner( this );
    _chart2_autoUpdateEvent_xjal.restoreOwner( this );
    _chart6_autoUpdateEvent_xjal.restoreOwner( this );
    _plot1_autoUpdateEvent_xjal.restoreOwner( this );
    _chart1_autoUpdateEvent_xjal.restoreOwner( this );
    _chart3_autoUpdateEvent_xjal.restoreOwner( this );
    _chart4_autoUpdateEvent_xjal.restoreOwner( this );
    _chart5_autoUpdateEvent_xjal.restoreOwner( this );
      finishReadObject_xjal( _stream, Main.class );
  }
```

Приложение В

ПРОГРАММА МОДЕЛИРОВАНИЯ ПРОЦЕССА ФОРМИРОВАНИЯ ГРАНУЛОМЕТРИЧЕСКОГО СОСТАВА ПРОДУКТОВ ИЗМЕЛЬЧЕНИЯ

Программный продукт предназначен для моделирования кинетики измельчения материала в струйной мельнице. В данном документе приводятся исходные коды встроенного программного обеспечения (ПО), реализованные с помощью компилятора JAVA.

Целью данного приложения является разработка имитационной модели кинетики измельчения в струйной мельнице, основанной на методе системной динамики моделирования стохастических динамических процессов. Разработанное ПО позволяет выполнять компьютерные эксперименты над моделью и проводить оптимизацию технологических параметров схемы измельчения относительно заданного критерия.

Для достижения поставленной цели решены следующие задачи:

1. Проведение структурного анализа процессов. Проводится формализация структуры процессов в струйной мельнице путем разложения его на подпроцессы, выполняющие определенные функции и имеющие взаимные функциональные связи.

2. Разработка формализованного описания модели. Графическое изображение имитационной модели, разработка функций, выполняемых каждым подпроцессом, условия взаимодействия всех подпроцессов и особенности поведения моделируемого процесса (временная, пространственная динамика) описаны на языке программировании для последующей трансляции.

3. Программная реализация модели: выбор средств моделирования, разработка методики моделирования, планирование имитационного эксперимента, выполнение имитационного моделирования, анализ и обобщение результатов.

Программное обеспечение, разработанное в данном приложении, представляет собой выполняемое приложение под ОС Windows. Оно выполнено при помощи среды программирования AnyLogic, использует язык программирования JAVA и специализированную объектно-ориентированную библиотеку объектов системной динамики. Данное ПО состоит из двух основных форм – формы презентации эксперимента и формы презентации модели

Презентация эксперимента

Первым окном программы является окно презентации эксперимента , которое автоматически отображается перед запуском презентации модели. Изначально оно показывает презентацию запущенного эксперимента. В этот момент модель еще не создана и создан только сам эксперимент. В этом окне инициализируются переменные модели, устанавливается модельное время, определяются применяемые численные методы и точность расчетов указанные при настройке эксперимента. На рис. В1 показано окно презентации эксперимента, на рис. В.2 окно настройки эксперимента.

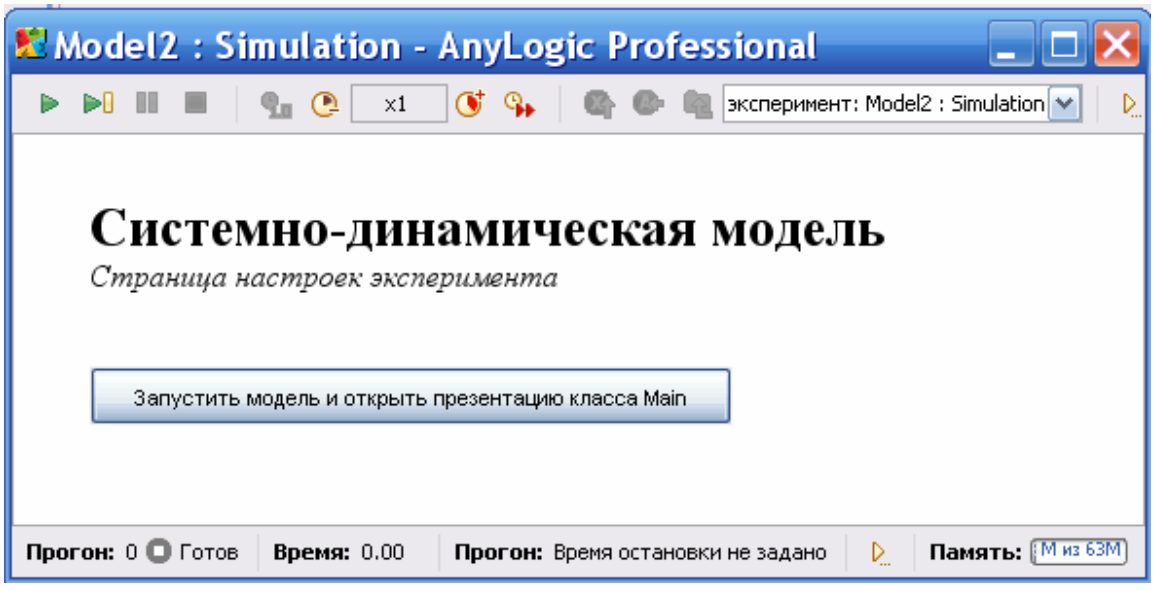


Рисунок В.1 – Окно презентации эксперимента

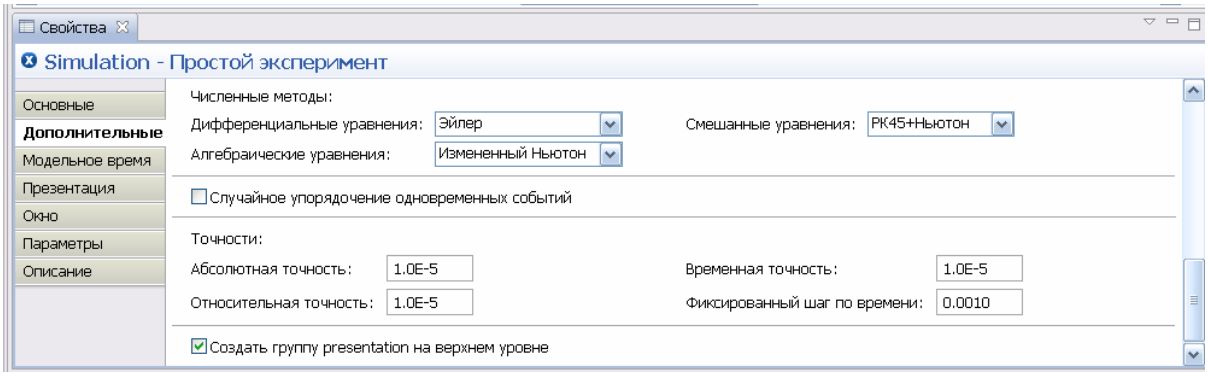


Рисунок В.2 – Окно настройки эксперимента

Презентация модели

Окно презентации модели(см.рис 3.3) состоит из двух панелей - панели управления и панели состояния и области отображения модели мельницы . Панель управления находится в верхней части окна презентации, а панель состояния - в нижней. Панель управления содержит секции с различными командами, позволяющими управлять выполнением модели, управлять обрисовкой презентации модели , осуществлять быструю навигацию по объектам модели и т.д. Основные команды панели управления и состояния приведены в табл. В1.

Описание модели

В данном приложении описана вариант программы 4-х фракционной модель исходного материала при измельчении. При этом на измельчение поступают три верхних фракции, а готовым продуктом измельчения является нижняя четвертая фракция, появляющаяся в результате измельчения первых трех.

Имитационная модель измельчения строится из следующих допущений о кинетике измельчения в струйной мельнице

Переменные модели

Переменными модели являются следующие параметры :

1. Интервал времени дискретной подачи исходного продукта в мельницу. Обозначение – Δt. Эта величина задается в миллисекундах.

2. Объем исходного материала подаваемого на измельчение в струйную мельницу. Обозначение – V.,Эта величина определяет объем материала поступающего в мельницу. Измеряется в кубических сантиметрах.

3. Фракционный состав материала подаваемого на измельчение в струйную мельницу. Соотношение между тремя фракциями материала подаваемого на измельчение.

Логическая схема модели

Имитационная модель кинетики переходов между фракциями материала внутри мельницы построена по следующему принципу:

–Имеется 4 накопителя F1, F2, F3, F4 имитирующих объем четырех фракций внутри мельницы. В начальный момент времени в накопителях F1, F2, F3 находится определенный объем исходного материала трех разных фракций; в накопителе F4 материал отсутствует.

–Имеется 3 накопителя имитирующих объем первых трех фракций внутри мельницы через предельно малое приращение времени – dt .

–Между накопителями F1→ F11, F2 → F22, F3 → F33 определены потоки соответствующих фракций материала F1_F1, F2_F2, F3_F3. Эти потоки определяют скорость изменения объема соответствующих фракций в мельнице.

–Между накопителями F1→ F11, F2 → F22, F3 → F33 также определены потоки соответствующих фракций материала F1_obr, F2_ F1_obr, F3 _obr. Эти потоки определяют циркуляцию соответствующих фракций внутри мельницы.

–Между накопителями F1, F2, F3, F4 также определены потоки измельченного материала:

–поток F1→F2 определяет поток второй фракции материала, возникающий в результате измельчения первой фракции за время dt ;

–поток F1→F3 определяет поток третей фракции материала, возникающий в результате измельчения первой фракции за время dt ;

–поток F1→F4 определяет поток четвертой фракции материала, возникающий в результате измельчения первой фракции за время dt ;

–поток F2→F3 определяет поток третей фракции материала, возникающий в результате измельчения второй фракции за время dt ;

поток F2→F4 определяет поток четвертой фракции материала, возникающий в результате измельчения второй фракции за время dt ;

поток F3→F4 определяет поток четвертой фракции материала, возникающий в результате измельчения третьей фракции за время dt ;

Значение накопителей F1, F2, F3 и F4 в каждый момент времени вычисляется в соответствии с дифференциальными уравнениями, правые часть которого составлена следующим образом:

$d(F1)/dt = -k_{11}*F11-k_{12}*F1\_F2-k_{13}*F1\_F3-k_{14}*F1\_F4$

$d(F2)/dt = k_{12}*F1-k_{22}*F22-k_{23}*F2\_F3-k_{24}*F2\_F4$

$d(F3/dt) = k_{12}*F1-k_{22}*F22-k_{23}*F2\_F3-k_{24}*F2\_F4$

$d(F4/dt) = F1\_F4 + F2\_F4 + F3\_F4$

Решение этих дифференциальных уравнений осуществляется численным методом Эйлера с начальным значением dt = 1 мсек.

На рис В.3 показано окно презентации модели, в котором отображена модель кинетики измельчения материала в терминах системной динамики и временной график,отображающий зависимости изменения грансоства по времени.

Значение коэффициентов *k,* определяющих изменения грансостава, в общем случае зависят от многих параметров, в том числе и времени измельчения и могут быть определены только экспериментальным путем. В данном дипломном проекте они определены только как функция от соответствующих накопителей и определяются долей от соответствующих накопителей. Значения соответствующих коэффициентов *k* приведены в таблице В.2.

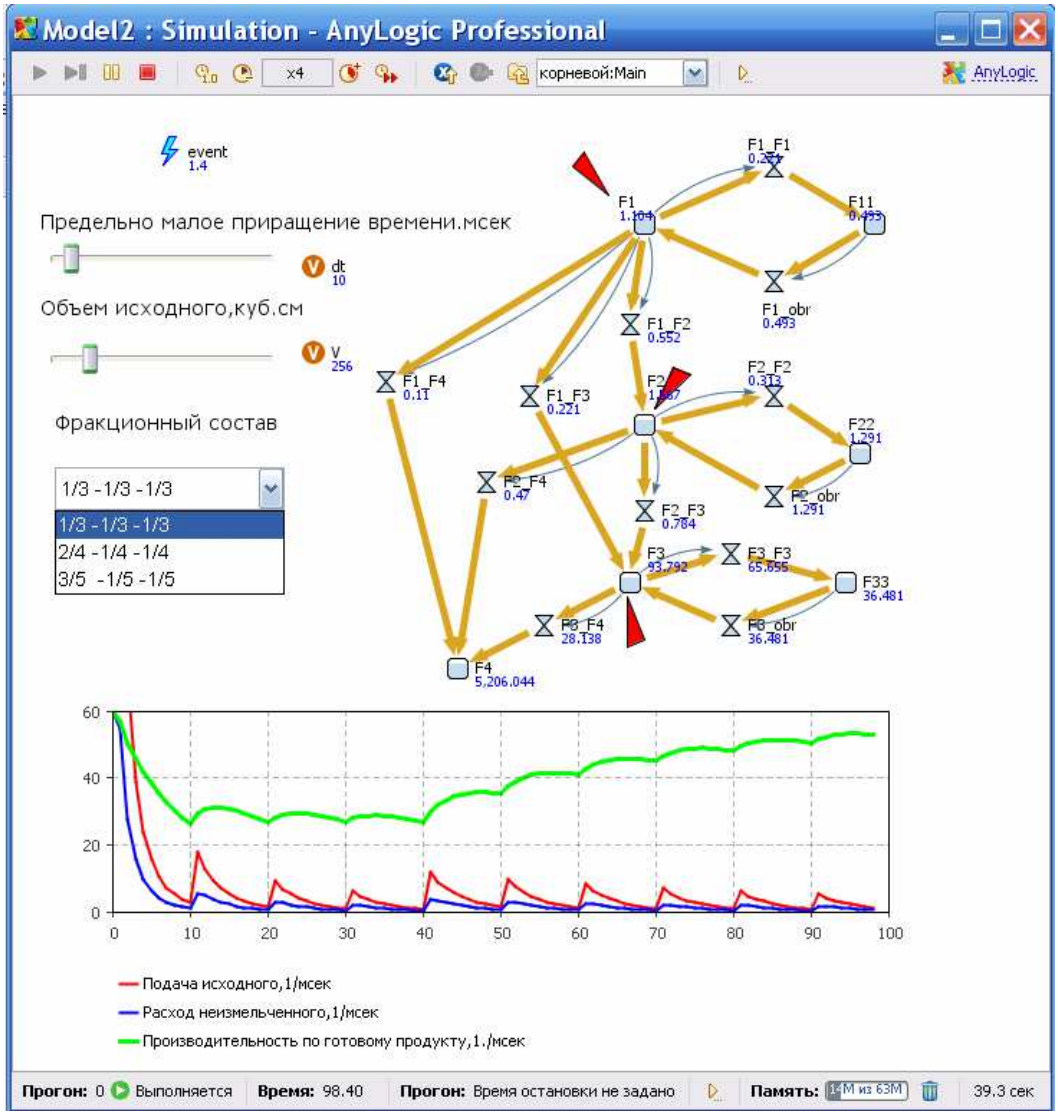


Рисунок В3 –Окно презентации модели

Таблица В.2 – Значения коэффициентов, определяющие изменения грансостава

	F1	F2	F3
F1	0.2	-	-
F2	0.5	0.2	-
F3	0.2	0.5	0.7
F4	0.1	0.3	0.3

Приложение Д

ПРОГРАММНЫЙ КОМПЛЕКС ОБРАБОТКИ ДАННЫХ АКУСТИЧЕСКОГО МОНИТОРИНГА СТРУЙНОГО ИЗМЕЛЬЧЕНИЯ

Разработанное программное обеспечение (ПО) в данном приложении является первой частью программного комплекса системы контроля качества продуктов струйного измельчения. Для функционирования программного комплекса необходимо получение данных с АЦП в режиме реального времени. Данные поступают с датчика, установленного на трубопроводе на пути движения готового измельченного продукта из классификатора в циклон. Описанная система предназначена для передачи и преобразования аналогового акустического сигнала, поступающего с датчика, установленного на выходе из классификатора струйной мельницы.

Программное обеспечение разработано совместно с ассистентом кафедры ПЗКС ГВУЗ «Национальный горный университет» Новодрановой Н.

Программное обеспечение разрабатывалось и тестировалось для устройства Л-Кард E14-440. Это универсальный модуль АЦП/ЦАП с интерфейсом USB 2.0, который особенно удобен для создания портативных измерительных систем на базе ноутбука. Однако ПО может быть адаптировано и для других аналогово-цифровых преобразователей (АЦП). И, по сути, является универсальным ПО для сбора данных с АЦП и передачи их для дальнейшей обработки, главная ценность которого заключается в возможности получения данных непосредственно в момент их поступления с какого-либо измерительного устройства или прибора.

Связь модуля с компьютером осуществляется через USB-интерфейс. USB (Universal Serial Bus) — универсальная последовательная шина, которая является на сегодняшний день широко распространенным промышленным стандартом расширения архитектуры персонального компьютера. Интерфейс USB позволяет осуществлять обмен информацией между хост-компьютером и множеством различных одновременно доступных периферийных

устройств, обеспечивая при этом возможность работы на трёх различных скоростях:

- низкая скорость(Low Speed– LS) – 1,5 Мбит/с(спецификация USB 1.1);
- полная скорость(Full Speed– FS) – 12 Мбит/с(спецификация USB 1.1);
- высокая скорость(High Speed– HS) – 480 Мбит/с(спецификация USB 2.0).

Фактически интерфейс соединяет между собой хост-контроллер USB и периферийные устройства. Хост-контроллер USB находится внутри персонального компьютера (в сущности, он является программно-аппаратной подсистемой персонального компьютера) и полностью контролирует работу всего интерфейса, т.е. все передачи данных по интерфейсу инициируются именно хост-контроллером. Для того чтобы к одному порту USB можно было подключать более одного устройства, применяются хабы (hub– устройство, обеспечивающее подключение к интерфейсу других устройств). Корневой хаб(root hub) находится внутри компьютера и подключен непосредственно к хосту. В интерфейсе USB используется специальный термин "функция"– это логически законченное устройство, выполняющее какую-либо специфическую функцию. В нашем случае это и есть модуль E14-440.

Всего в интерфейсе USB может быть использовано четыре типа пересылок информации, однако в модуле E14-440 из всего этого набора используются только управляющие и потоковые пересылки.

В связи с тем, что в интерфейсе USB реализован довольно сложный протокол обмена информацией, в устройстве сопряжения с интерфейсом USB необходимо применять микропроцессорный блок, обеспечивающий полную поддержку протокола. В качестве такового на модуле E14-440 используется согласованная связка микросхем USB интерфейса PDIUSBD12 от фирмы NXP Semiconductors и микроконтроллера AVR AT90S8515 или ATmega8515 от фирмы Atmel Corporation. При наладке модуля E14-440 в ППЗУ микроконтроллера AVR зашивается специально разработанный драйвер, одна из основных задач которого— обеспечить корректный интерфейс модуля с хост-компьютером. Помимо набора стандартных

запросов, которые поступают с хост-компьютера и должны пониматься всеми без исключения USB-устройствами, в модуле E14-440 организован целый ряд специализированных запросов ‘Vendor Request’ для целей реализации требуемых алгоритмов функционирования модуля. Также как и для всех стандартных запросов, для посылки запроса такого рода используется управляющая пересылка. Каждый специализированный запрос имеет свой уникальный номер и по мере поступления в AVR обрабатывается соответствующим образом, т.е. AVR выполняет действия, однозначно предопределенные номером запроса. Например, в ответ на поступление V_RESET_MODULE запроса, микроконтроллер AVR просто осуществляет сброс модуля. С программной точки зрения посылка из хост-компьютера требуемого запроса типа ‘Vendor Request’ в модуль E14-440 осуществляется с помощью обычной Windows API функции DeviceIoControl().

Стандартная Windows API функция DeviceIoControl является одной из самых ключевых функций в деле организации взаимодействия приложения в PC с модулем E14-440. Её следует применять для передачи управляющих кодов непосредственно в драйвер с дескриптором (идентификатором). Управляющие коды определяют те действия драйвера, которые он должен выполнить. Для модуля E14-440 в драйвере предусмотрены три управляющих кода:

1. DIOC_SEND_COMMAND. Данный код предписывает драйверу выполнить соответствующую управляющую пересылку, т.е. посылку в модуль запроса типа‘Vendor Request’ с требуемым номером.
2. DIOC_RESET_PIPE1. Данный код предписывает драйверу выполнить сброс канала, обеспечивающего потоковые пересылки для записи данных в модуль.
3. DIOC_RESET_PIPE3. Данный код предписывает драйверу выполнить сброс канала, обеспечивающего потоковые пересылки для чтения данных из модуля.

Помимо ответственности за организацию и поддержание в

надлежащем виде USB-интерфейса модуля E14-440 с хост-компьютером, на микроконтроллер AVR ложится довольно непростая задача по корректному взаимодействию с цифровым сигнальным процессором, расположенным на модуле.

Драйвер микроконтроллера AVR написан таким образом, что разрешает пользователю из приложения в PC путём посылки в модуль соответствующих запросов USB задавать функциональное состояние модуля E14-440. При этом аппаратно модуль реализован так, что вся периферия (АЦП, ЦАП, ТТЛ линии и.т.д.) находится исключительно под управлением цифрового сигнального процессора. Поэтому от AVR настоятельно требуется обладать возможностью оперативного вмешательства в любой момент времени в текущую работу драйвера DSP без остановки в его функционировании (кроме сброса DSP).

В соответсвии с составом выполняемых системой функций основной задачей программы является сбор данных с аналого-цифрового преобразователя, фиксирующего акустические сигналы. Поступление этих сигналов с АЦП в реальном масштабе времени должно учитывать следующие параметры:количествово активных каналов; частоту ввода данных; количество отсчетов в запросе DataStep (кратное 32);столько блоков по DataStep отсчётов нужно собрать в файл данных.

При сборе данных использовалась библиотека Lusbapi.dll, описывающая интерфейс модуля E14-440, и поставляемая совместно с АЦП.

Следующей задачей программы является формирование данных для дальнейшей обработки и анализа. В этой части ПО установлены необходимые параметры ввода данных с модуля E14-440 и разрешена корректировка данных на уровне драйвера DSP. В программе предусмотрен обычный сбор данных безо всякой синхронизации ввода и использованы фирменные калибровочные коэффициенты, которые храняться в ППЗУ модуля E14-440. Задачей программы было моделирование входного сигнала, поступающего на АЦП.

При создании программного обеспечения системы обработки данных акустического мониторинга работы струйной мельницы используется среда разработки Embarcadero Delphi 2010.

Фактически работа программного обеспечения состоит из нескольких этапов: инициализация модуля, потоковый ввод данных, обработка данных и корректировка данных с использованием корретировочных коэффициентов.

Инициализация модуля включает несколько действий, выполняемых в следующей последовательности: сброс флагов ошибки потока ввода и завершённости потока сбора данных. Пока открытого файла нет, сбрасываются счётчики инициализации, очищаются дескрипторы ввода и вывода, очищается экран дисплея, проверяется версия используемой DLL библиотеки. Затем необходимо получить указатель на интерфейс для модуля E14-440, получить идентификатор устройства и прочитать название модуля в текущем виртуальном слоте. Следующим этапом нужно получить скорость работы шины USB и подключить код драйвера DSP из соответствующего ресурса DLL библиотеки Lusbapi.dll. После чего проверяется загрузка модуля и определяется номер версии загруженного драйвера DSP. Завершает инициализацию получение текущих параметров ввода данных, установка желаемых параметров ввода данных с модуля E14-440 и разрешение корректировки данных на уровне драйвера DSP.

Перед началом ввода данных необходимо выделить нужное количество памяти под буфер данных и создать файл для записи собранных данных. После чего запусткается поток сбора данных и отображаются параметры работы модуля по вводу данных на экране монитора.

Основной цикл программы включает ожидание окончания сбора данных, окончание работы потока ввода данных, а также просмотр ошибок при сборе данных. Потоковая функция ReadThread начинается с остановки работы АЦП и сброса USB-канала чтения данных.Затем формируются необходимые для сбора данных структуры, создаются объекты ядра ОС «событие» для асинхронного запроса на ввод данных, начинается первый асинхронный сбор

данных в буфер. После чего можно запускать цикл сбора данных.В каждой итерации цикла выполняется запрос на очередную порцию вводимых данных и поток переходит в ожидание выполнения очередного запроса на сбор данных.В созданный заранее файл записывается очередная порция данных, после чего увеличивается счётчик полученных блоков данных. Так продолжается до поступления последней порции данных.

Корректировка данных выполнялась с учетом корректировочных коэффициентов смещения нуля и корректировочных коэффициентов масштаба на уровне драйвера DSP. Для проверки правильности работы программы целесообразно использовать моделирование входного сигнала, поступающего на АЦП. Для этой цели используется программа, генерирующая сигналы различной звуковой частоты. Т.е. сигнал со звуковой карты компьютера поступает на вход АЦП, а программа сбора данных записывает значения, получаемые на выходе АЦП и строит график.

Одной из самых ключевых функций, использующихся при организации взаимодействия приложения в PC с модулем E14-440 является стандартная Windows API функция DeviceIoControl( ), которая отправляет управляющий код непосредственно указанному драйверу устройства, заставляя соответствующее устройство выполнить соответствующую операцию.

Для асинхронных операций, DeviceIoControl возвращает значение немедленно, а объект события подает сигнал, когда операция завершается. В противном случае, функция не возвращает значение до тех пор, пока операция не завершится или не произойдет ошибка.

Также в разработанном ПО присутсвуют следующие функции:

процедура InitConsoleHandles - инициализация глобальных переменных ввода/вывода; функция WaitingForRequestCompleted (var ReadOv: OVERLAPPED): boolean - ожидание завершения выполнения очередного запроса на сбор данных; Функция ReadThread (var param: pointer): DWORD – фукция, запускаемая в качестве отдельного потока для сбора данных с модуля E14-440.

Аварийное завершение программы: procedure TForm1.AbortProgram(ErrorString: string; AbortionFlag : bool = true);

Отображение ошибок возникших во время работы потока сбора данных: procedure TForm1.ShowThreadErrorMessage;

Преобразование строки: function StrPas2(s: array of byte): string;

Обработка события нажатия кнопки «Начать сбор данных»: procedure TForm1.Button1Click(Sender: TObject);

Запуск программы производится с жесткого диска или любого другого носителя данных позволяющего производить запись на его поверхность. Запуск программ для моделирования входного сигнала производится запуском исполняемых файлов Gen.exe и ACP.exe из директории программы.

Входными данными для данной системы являются:

- тип акустического сигнала;
- частота акустического сигнала в Гц;
- амплитуда акустического сигнала в миливольтах;

Выходными данными для данной системы являются:

- график преобразованного сигнала A(t);
- массив данных для дальнейшей обработки.

Для тестирования правильности работы разрабатываемого программного обеспечения используется программа генерации звукового сигнала и программа сбора данных с аналого-цифрового преобразователя.

Данная программа позволяет задавать различные параметры генерации звукового сигнала:

- вид сигнала (синусоида, прямоугольный, треугольный, пилообразный, белый шум). Для задания разновидности сигнала необходимо выбрать нужный пункт в разделе «Вид сигнала»;
- частоту сигнала. Пользователю предоставляется возможность задавать частоту в герцах, килогерцах и *100Гц;
- амплитуду звукового сигнала (мВ).

Задание последних двух параметров осуществляется передвижением мышью ползунка до нужного значения в соответсвующем разделе программы. После того как сгенерирован звуковой сигнал необходимо запустить программу сбора данных и нажать кнопку «Начать сбор данных».

Рисунок Д1 – Программа генерации звукового сигнала.

Каждый раз после изменения параметров генерируемого сигнала необходимо нажимать кнопку «Начать сбор данных» для того, чтобы на графике отобразились изменения.

При этом в нижней части экрана выводятся параметры работы модуля по вводу данных, а в сетке выводится график, отображающий получаемые с АЦП амплитуды сигналов.

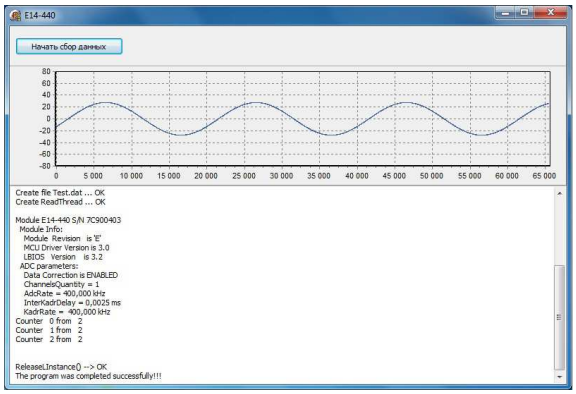


Рисунок Д3 – Окно программы сбора данных. Синусоида, 20Гц, 25мВ.

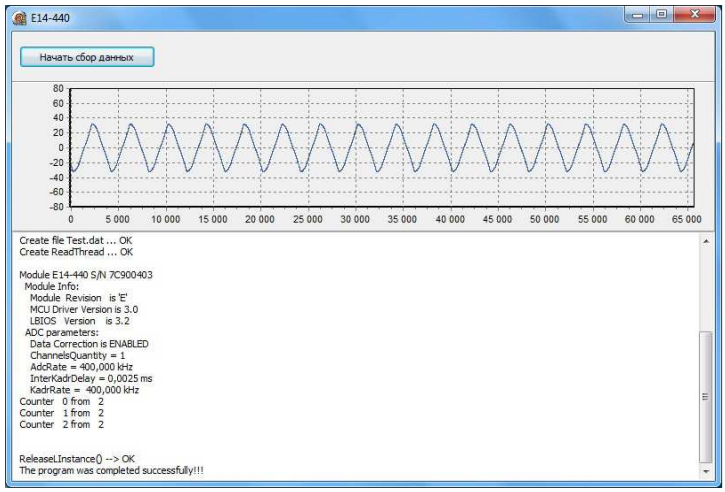


Рисунок Д4 – Окно программы сбора данных. Треугольный, 100Гц,

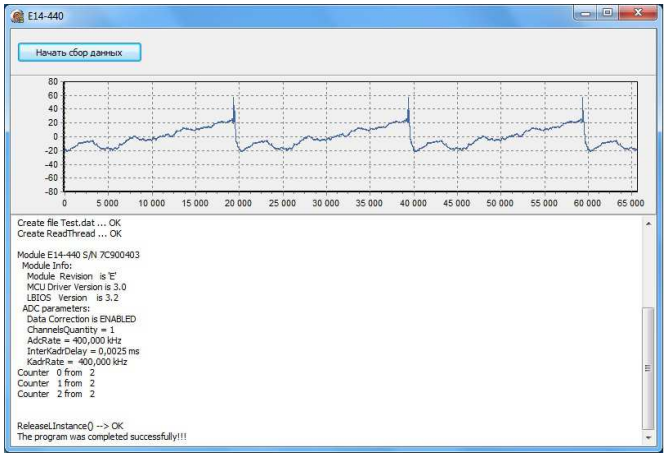


Рисунок Д5 – Окно программы сбора данных. Пилообразный, 20Гц, 60мВ.

Особенность разработанного ПО состоит в возможности получения данных непосредственно в ходе технологического процесса и передачи их для дальнейшей обработки без использования ПО, идущего в комплекте с АЦП.

Приложение Е

ПРОГРАММА ОЦЕНКИ ДИСПЕРСНОСТИ ПРОДУКТОВ ПО ДАННЫМ АКУСТИЧЕСКОГО МОНИТОРИНГА

Разработанное программное обеспечение предназначено для контроля дисперсности продукта измельчения по величине амплитуды акустического сигнала, что позволяет своевременно зафиксировать наличие брака сразу после выхода измельчаемого материала из классификатора.

Приложение разработано на языке Object Pascal в среде Delphi 2010 с использованием языка структурированных запросов SQL. Среда разработки Embarcadero Delphi 2010 отличается быстродействием, компактностью, полнофункциональным пользовательским интерфейсом а также способностью интегрироваться едва ли не к любой базе данных, не требуя дополнительной доработки.

Главное окно приложения представляет собой форму, содержащую следующие элементы: область построения графика по данным текущего помола, поля для выбора из списка материала и заданной дисперсности, поля для внесения частоты вращения ротора и временных интервалов при просмотре фрагмента графика, кнопки для запуска и остановки процесса, кнопки для удаления и сохранения данных.

Меню приложения включает следующие пункты

- Обработка данных (с*охранить данные*, *удалить данные*, *просмотр архива);*
- Справочники (*материалы*, *причины хранения*, *нормы)*
- Архив данных

Входными данными для данной системы являются: материалы; причины хранения данных; нормы амплитуд АС; заданная дисперсность; получаемые во время работы программы значения амплитуд АС; значения временных интервалов. Выходными данными для данной системы являются: график АС(t); сохраненные данные.

На рис. Е1 представлена схема базы данных, на рис. Е2 – главное окно программы.

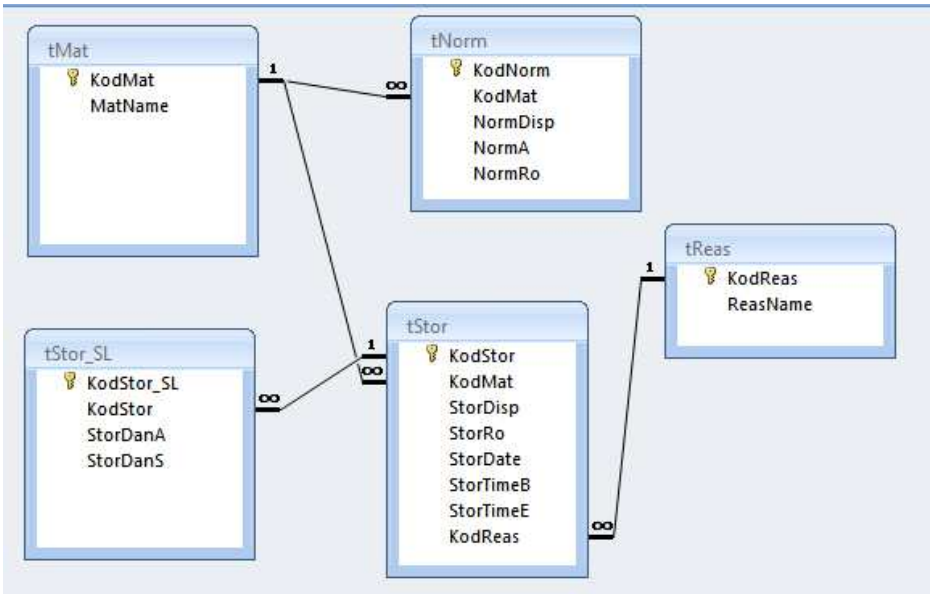


Рисунок Е1. – Схема базы данных

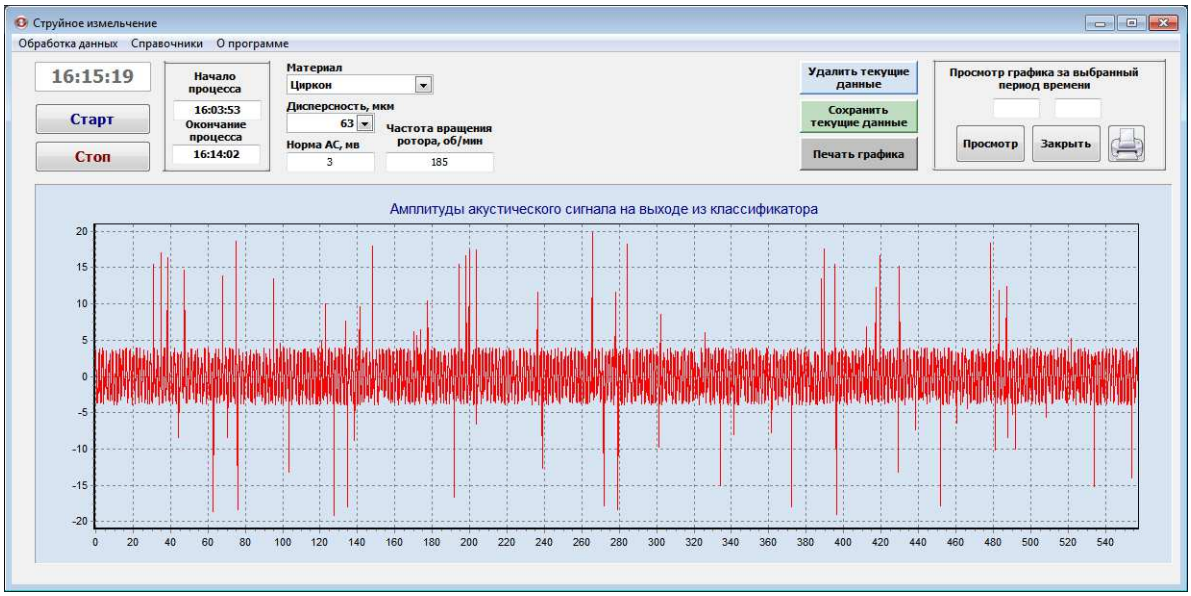


Рисунок Е2. – Главное окно программы.

Данные не удаляются автоматически после закрытия приложения, поэтому, если после предыдущего цикла помола данные не были удалены, то после запуска приложения пользователь увидит график, построенный по

предыдущим данным. В случае необходимости он имеет возможность сохранить данные, просмотреть заданные участки графика и выполнить экспорт.

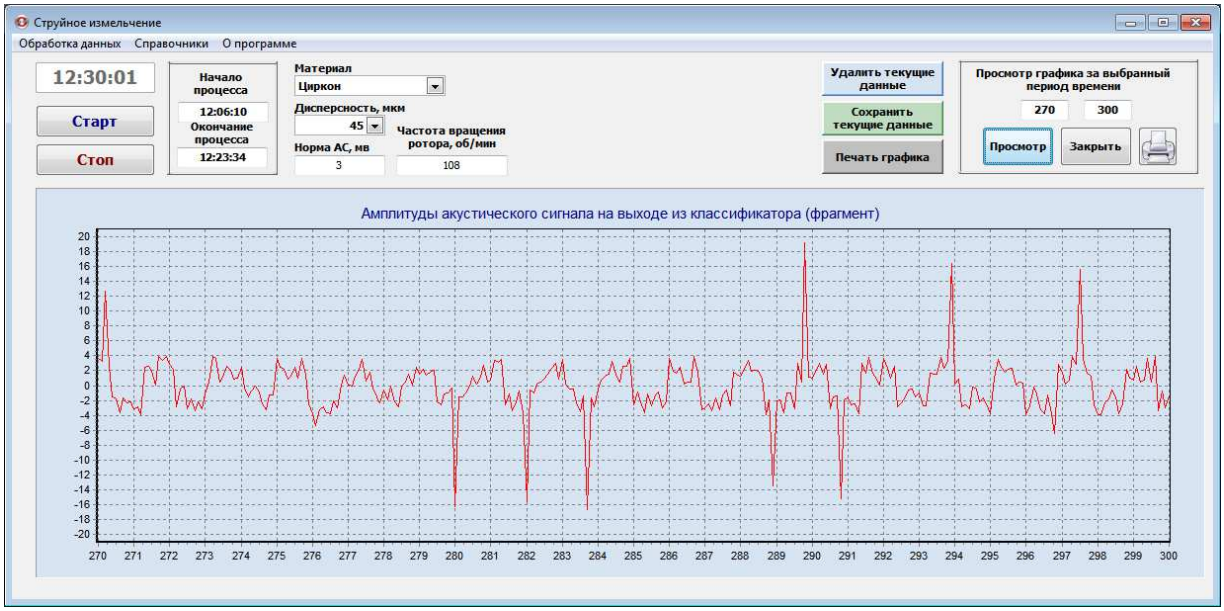


Рисунок Е3. – Просмотр фрагмента графика.

Как только программа зафиксирует большое количество сигналов, амплитуды которых значительно превышают норму, то пользователю выдается сообщение и в верхней части главного окна приложения появляется надпись и мигающая пиктограмма ярко желтого цвета с восклицательным знаком.

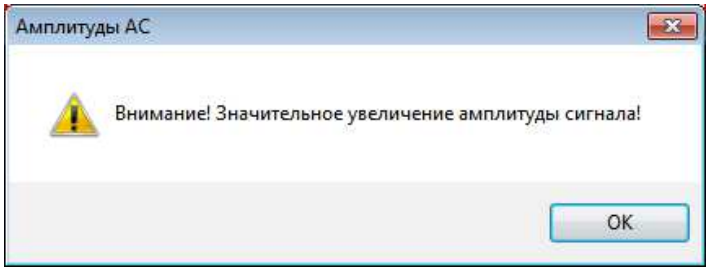


Рисунок Е4. – Сообщение о возникновении критической ситуации.

Для остановки процесса записи амплитуд сигналов необходимо нажать кнопку «Стоп». Поля *Начало процесса* и *Окончание процесса* заполняются

текущим временем при нажатии на кнопки «Старт» и «Стоп» соответственно.

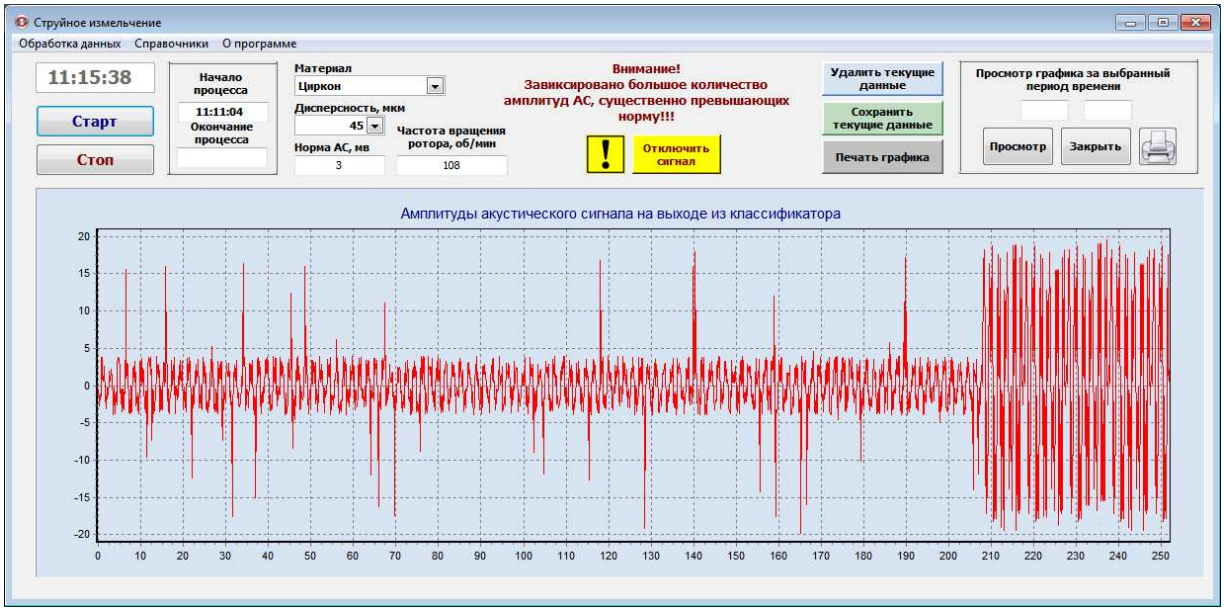


Рисунок Е5. – Главное окно программы. Сигналы, оповещающие пользователя о возникновении критической ситуации.

Для сохранения данных форма «Сохранение и экспорт данных» предоставляет возможность сохранить все данные текущего цикла помола, либо часть данных, соответствующих заданному временному интервалу, а также произвести экспорт данных в MS Excel.

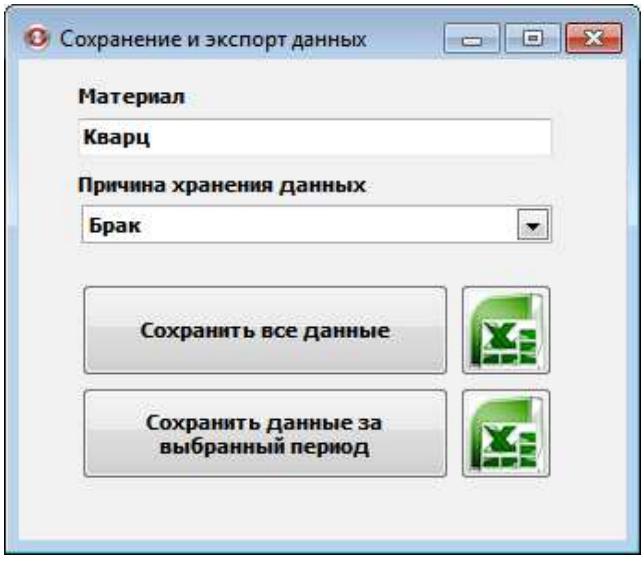
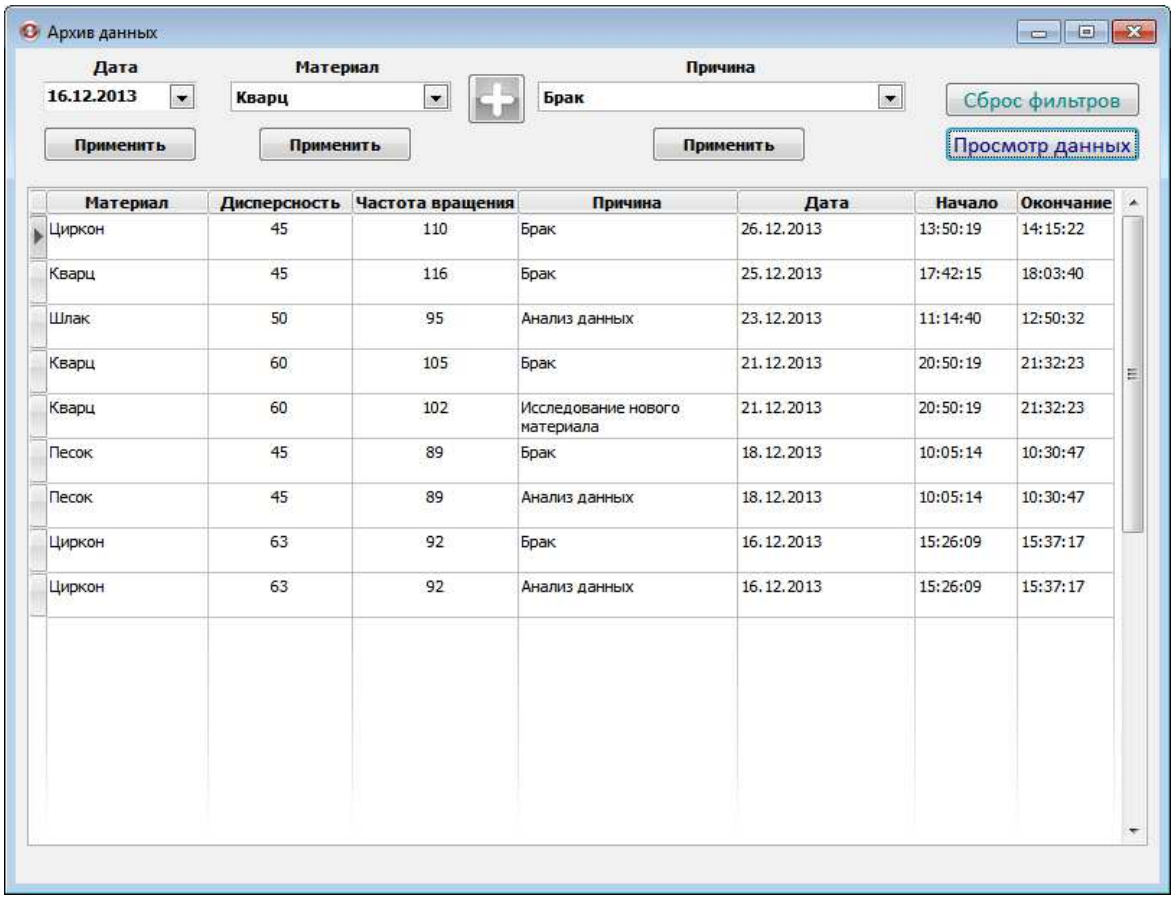


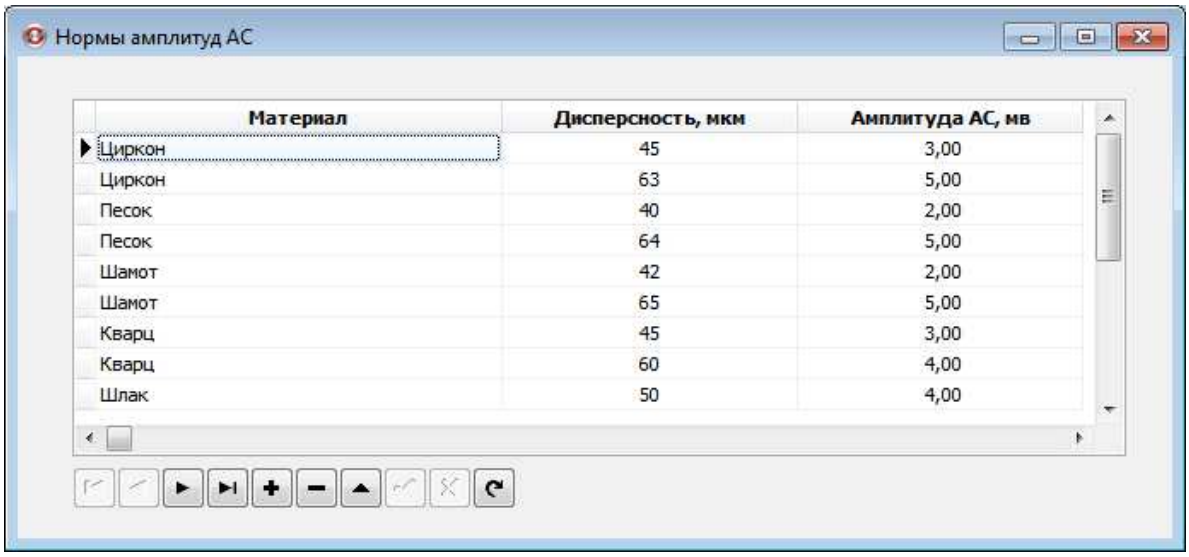
Рисунок Е6. – Форма «Сохранение и экспорт данных».

Для сохранения данных обязательно должна быть указана причина из списка *Причина хранения данных*. Все сохраненные данные можно просмотреть. Для этого необходимо открыть форму «Архив данных», выбрав пункт меню *Обработка данных* → *Просмотр архива.*



Материал	Дисперсность	Частота вращения	Причина	Дата	Начало	Окончание
Циркон	45	110	Брак	26.12.2013	13:50:19	14:15:22
Кварц	45	116	Брак	25.12.2013	17:42:15	18:03:40
Шлак	50	95	Анализ данных	23.12.2013	11:14:40	12:50:32
Кварц	60	105	Брак	21.12.2013	20:50:19	21:32:23
Кварц	60	102	Исследование нового материала	21.12.2013	20:50:19	21:32:23
Песок	45	89	Брак	18.12.2013	10:05:14	10:30:47
Песок	45	89	Анализ данных	18.12.2013	10:05:14	10:30:47
Циркон	63	92	Брак	16.12.2013	15:26:09	15:37:17
Циркон	63	92	Анализ данных	16.12.2013	15:26:09	15:37:17

Рисунок Е7 – Форма «Архив данных»



Материал	Дисперсность, мкм	Амплитуда АС, мв
Циркон	45	3,00
Циркон	63	5,00
Песок	40	2,00
Песок	64	5,00
Шамот	42	2,00
Шамот	65	5,00
Кварц	45	3,00
Кварц	60	4,00
Шлак	50	4,00

Рисунок Е8 – Форма «Нормы амплитуд АС».

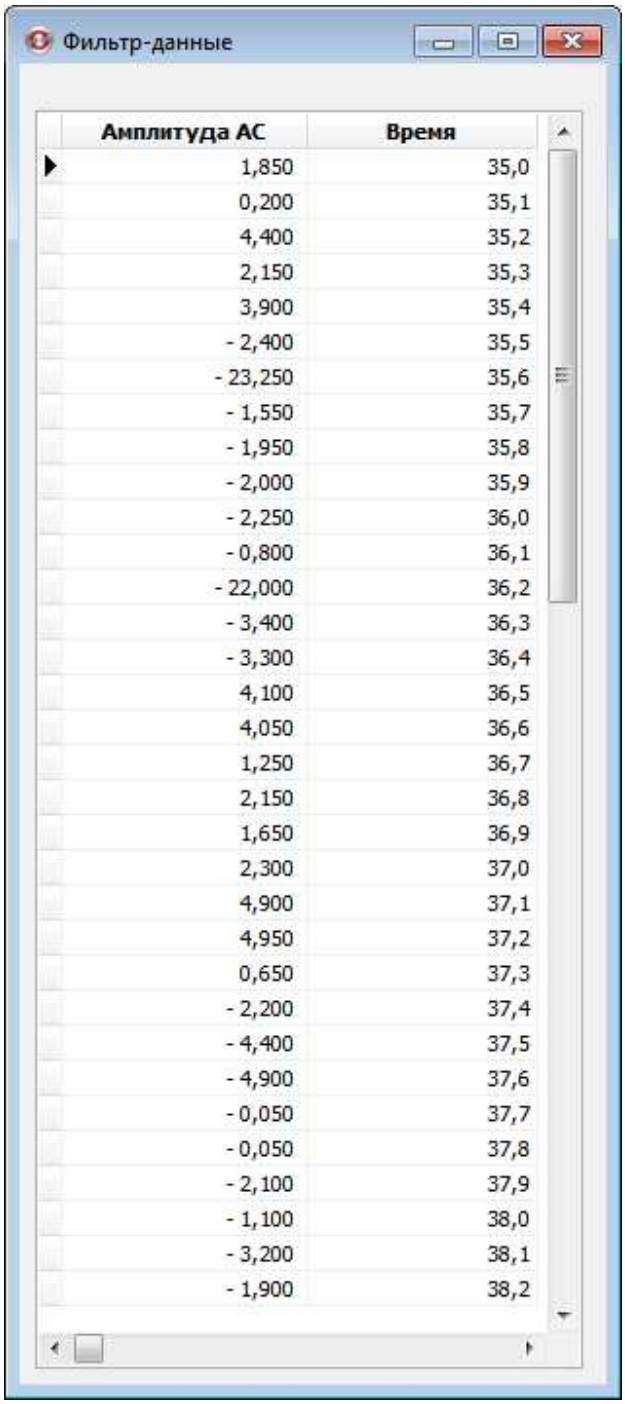


Рисунок Е9 – Форма «Фильтр-данные».

Все данные, использующиеся в программе для выбора значений из списков, хранятся в соответствующих справочниках, которые открываются при выборе пункта меню *Справочники*.

Приложение Ж

МЕТОДИКА АНАЛИЗА РАБОТЫ МЕЛЬНИЦЫ ПО СИГНАЛАМ АКУСТИЧЕСКОГО МОНИТОРИНГА

В данном приложении режимы работы струйной мельницы анализируются по выходному сигналу зоны измельчения струйной мельницы с использованием показателя Херста. Исследованы следующие методы определения параметра Херста: R/S-анализ (метод нормированного размаха) и изменение во времени дисперсии агрегированного ряда, проведен сравнительный анализ статистических характеристик оценок, полученных этими методами.

Известно, что самоподобный объект - объект, в точности или приближённо совпадающий с частью себя самого. Компактное топологическое пространство X самоподобно, если существует конечное множество S, индексирующее набор отображений $\{f_S\}_S \in_S$, для которых

$$X = U_{s \in S} f_S(X).$$

Если $X \subset Y$, то X называется самоподобным, если оно является единственным непустым подмножеством Y, для которого вышеприведённое уравнение выполняется при заданном семействе $\{f_S\}_S \in_S$. В таком случае $\Im = (X, S, \{f_S\}_S \in_S)$ именуется самоподобной структурой.

Стохастический процесс $X(t)$ является статистически самоподобным, если процесс a-$HX(at)$ обладает теми же статистическими характеристиками второго порядка, что и $X(t)$. Долгосрочная зависимость означает медленное (гиперболическое) убывание во времени автокорреляционной функции случайного процесса. Параметр H, называемый параметром Херста и представляет собой меру самоподобия или меру длительности долгосрочной зависимости стохастического процесса. Значение H = 0,5 указывает на отсутствие долгосрочной зависимости. Чем ближе значение H к 1, тем выше степень устойчивости долгосрочной зависимости.

Величина коэффициента H характеризует отношение силы тренда (детерминированный фактор) к уровню шума (случайный фактор).

Если показатель Херста принимает значения: $0<H<0.5$, то ряд является антиперсистентный временной ряд, то есть ряд, для которого более вероятна смена предыдущего направления. Антиперсистентный ряд так же называют «розовым шумом». Эти процессы наиболее характерны для эффектов турбулентности. $H = 0.5$ – временной ряд стохастичен. Такой процесс называют «белым шумом». $0.5<H<1$ – персистентный временной ряд (эти процессы еще называют «черным шумом»), то есть ряд, которому присуща трендовость (направленность).

Имеются три различных классификации для показателя Херста:

1) $H = 0.5$. Указывает на случайный ряд. События случайны и некоррелированны. Настоящее не влияет на будущее. Функция плотности вероятности может быть нормальной кривой, однако, это не обязательное условие. R/S-анализ может классифицировать произвольный ряд, безотносительно к тому, какой вид распределения ему соответствует.

2) $0 \leq H < 0.5$. Данный диапазон соответствует антиперсистентным, или эргодическим, рядам. Такой тип системы часто называют – «возврат к среднему». Устойчивость такого антиперсистентного поведения зависит от того, насколько H близко к нулю. Такой ряд более изменчив, чем ряд случайный, так как состоит из частых реверсов спад-подъем.

3) $0.5 <H<1.0$. Исследуемые ряды персистентные, или трендоустойчивые ряды. Если ряд возрастает (убывает) в предыдущий период, то, вероятно, что он будет сохранять эту тенденцию какое-то время в будущем. Чем ближе H к 0.5, тем более зашумлен ряд и тем менее выражен его тренд. Персистентный ряд – это обобщенное броуновское движение, или смещенные случайные блуждания. Сила этого смещения зависит от того, насколько H больше 0.5 [109].

Таким образом, показатель Херста позволяет проанализировать степень организованности процесса. При случайном, хаотическом процессе, когда нет никакой закономерности во временном ряде, показатель Херста равен 0,5. Если же ряд имеет некоторую закономерность, показатель Херста отличается от 0,5:

для персистентного ряда существует положительная корреляция между прошедшими и будущими событиями (показатель Херста больше 0,5), для антиперсистентного ряда наблюдается отрицательная корреляция между прошедшими и будущими событиями (показатель Херста меньше 0,5).

Для оценки возможности использования показателя Херста для акустических сигналов были проанализированы акустические сигналы, записанные при акустическом мониторинге различных режимов загрузки струйной мельницы. Анализировались следующие режимы функционирования струйной мельницы:

1. перед загрузкой (струи энергоносителя без материала) (рис. Ж1а);
2. подача материала и первые секунды измельчения (загрузка мельницы) (рис. Ж1б);
3. рабочий режим (процесс измельчения) (рис. Ж1в);
4. разгрузка (мельница почти пустая, заканчивается измельчение или же необходима повторная дозагрузка) (рис. Ж1г).

Сложность классификации сигналов акустического мониторинга заключается в том, что записанный акустический сигнал является нестационарным. Использование нескольких методов классификации акустических сигналов, сопровождающие процесс измельчения, позволяет более точно контролировать уровень внутримельничной загрузки мельниц.

К настоящему времени разработано множество методов оценки показателя Херста на основе эмпирических данных. Данные методы характеризуются различной точностью получаемых результатов. В настоящей работе используются следующие методы:

- метод нормированного размаха (R/S-анализ);
- метод изменения дисперсии агрегированного ряда

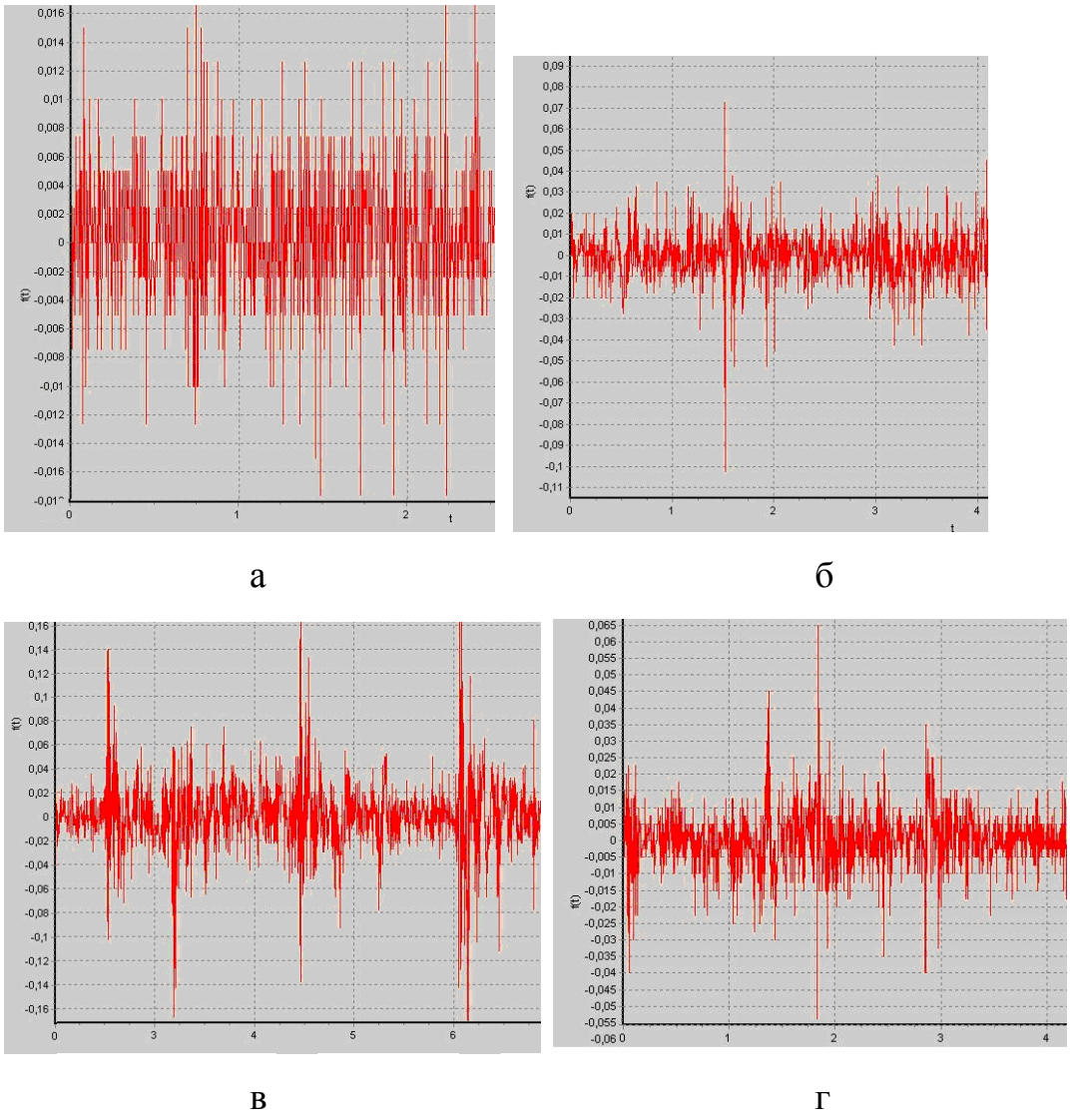


Рисунок Ж1 – Акустические сигналы перед загрузкой (а), при загрузке (б), рабочем режиме (в) и разгрузке (г)

Метод нормированного размаха

1. Пусть имеется нестационарный временной ряд длины M. Преобразуем его во временной ряд длины $N=M-1$;

2. Разделим период времени N на A смежных подпериодов длины n, так, что $A*n=N$. При этом N, A и n подбираются таким образом, чтобы $\frac{M-1}{n}$ было целочисленным значением. Пометим каждый подпериод I_a с учетом того что $a=1..N$. Введем двойной индекс – каждый элемент N_i в I_a

переобозначим $N_{k,a}$, $k=i$. Для каждого I_a длины n среднее по интервалу значение определяется:

$$e_a = (1/n) \sum_{k=1}^{n} N_{k,a};$$

3. Находим элементы временного ряда накопленных отклонений ($X_{k,a}$) от среднего значения для каждого подпериода I_a:

$$X_{k,a} = \sum_{i=1}^{k} (N_{i,a} - e_a);$$

4. Диапазон определяется как максимальное значение за вычетом минимального значения $X_{k,a}$ в пределах каждого подпериода I_a:

$$R_{Ia} = max(X_{k,a}) - min(X_{k,a}), 1 \le k \le n$$

5. Выборочное стандартное отклонение рассчитывается для каждого подпериода I_a:

$$S_{I_a} = \sqrt{(1/n) \sum_{k=1}^{n} (N_{k,a} - e_a)^2};$$

6. Каждый диапазон R_{Ia} нормализуется путем деления на соответствующее значение S_{Ia}. На шаге (2) мы получили смежные подпериоды длины n. Следовательно, среднее значение $(R/S)_n$ определяется:

$$(R/S)_n = (1/A) \sum_{a=1}^{A} (R_{I_a} / S_{I_a});$$

7. Длина n итеративно увеличивается до следующего более высокого значения. Шаги (2-6) повторяются до n=(M-1)/2. Получаем два вектора: $\overline{n}$ и $\frac{\overline{R}}{S}$.

8. Теперь, на основе значений, полученных на шаге (7), применяется регрессию к уравнению $(R/S)_n = c*n^H$, где $c = const$, H – показатель Херста. Предварительно логарифмируем обе части уравнения регрессии. Отрезок, отсекаемый на координатной оси, является оценкой log(c), константой. Наклон прямой является оценкой показателя Херста H.

Метод изменения дисперсии агрегированного ряда

Для агрегированных временных серий $x(m)$ самоподобного процесса дисперсия при больших значениях параметра m подчиняется следующей формуле:

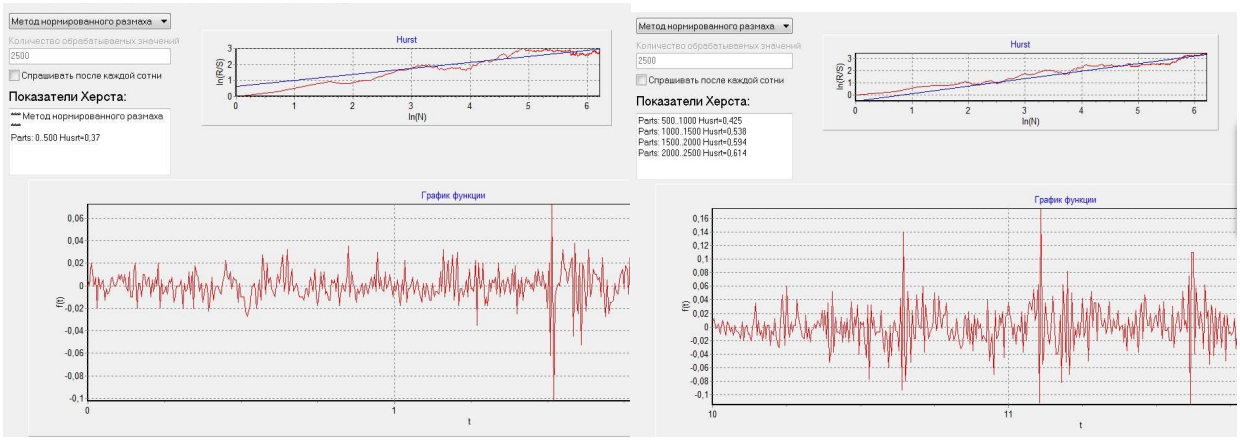
$$Var(x^{(m)}) \approx \frac{Var(x)}{m^{\beta}}$$

В этом случае параметр самоподобия $H=1-\beta/2$ можно определить, если сгенерировать агрегированный процесс на разных уровнях агрегации m и вычислить дисперсию для каждого уровня. График зависимости $log(Var(x^{(m)}))$ от $\log(m)$ будет представлять собой прямую линию с наклоном, равным $-\beta$.

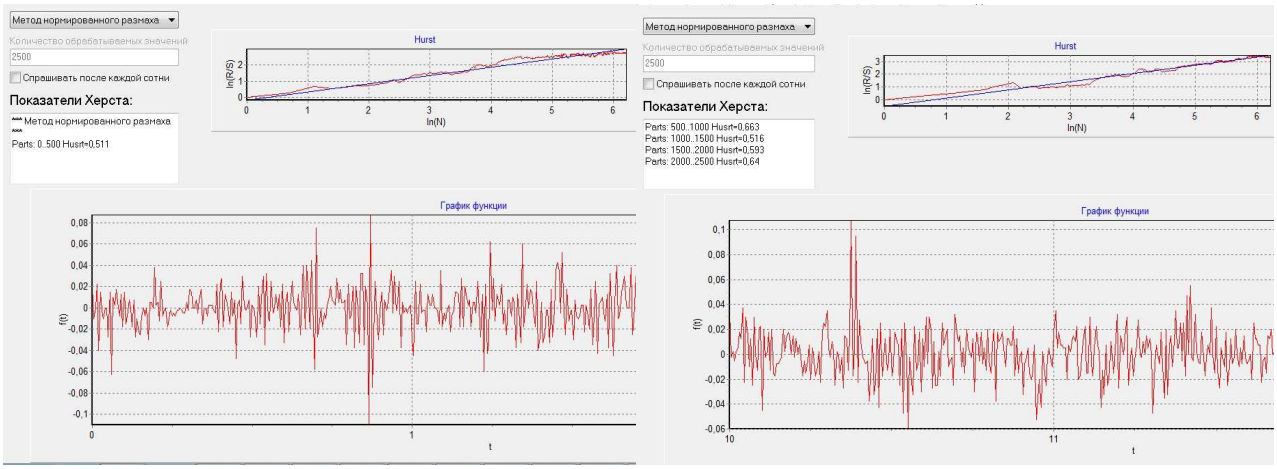
В ходе акустического мониторинга были получены сигналы для разных состояний мельницы:в рабочем режиме, во время разгрузки мельницы, когда мельница пустая и во время загрузки мельницы . Для удобства анализа параметров Херста сигналы были разбиты на 5 серий (частей) для состояния загрузки, рабочего режима и разгрузки мельницы, и на 3 серии для состояния, когда мельница пустая. На рис. Ж2 представлены результаты выполнения программы - обработка сигнала и нахождение параметра Херста для первый и последних серий сигналов методом нормированного размаха.

Аналогично были получены параметры Херста акустического сигнала струйной мельницы методом изменения дисперсии. На рис. Ж3 показаны результаты определения параметра H для последних серий различных режимов измельчения.

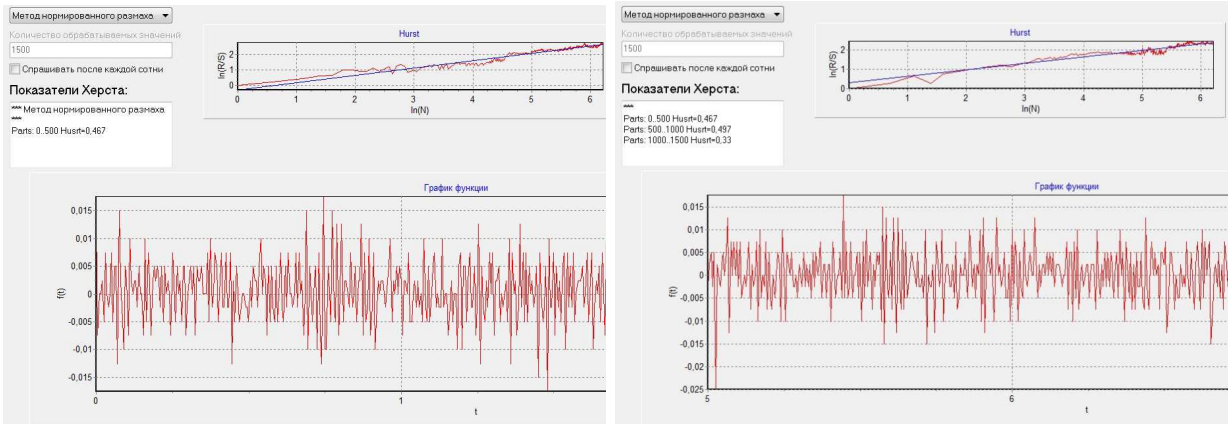
Значения H для исследуемых сигналов акустического мониторинга струйного измельчения шлака изменялись во всем возможном диапазоне $0 < H < 1$. Для полученного временного ряда рассчитывалась оценка H вышеописанными методами: R/S-анализа (*Hrs*) и дисперсии агрегированного ряда (*Hd*). Значения показателя Херста, полученные различными методами, приведены в таблице Ж1.



а



б



в

Рисунок Ж2 – Нахождение параметра Херста методом нормированного размаха для первой и последней части выходного сигнала при загрузке струй материалом (а), рабочем режиме измельчения (б) и полной разгрузке мельницы (струи без материала в).

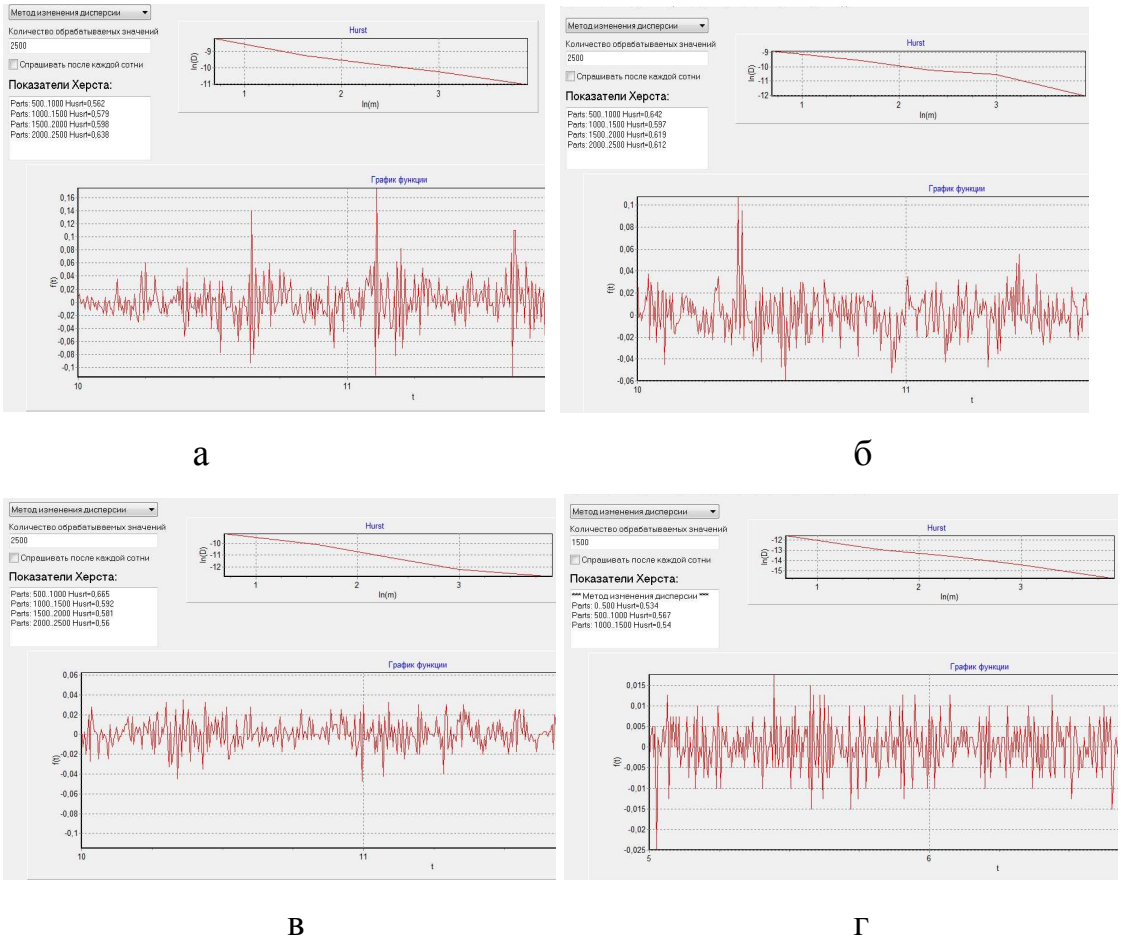


Рисунок Ж3 – Нахождение параметра Херста методом изменения дисперсии для пятой части выходного сигнала во время загрузки (а), рабочего режима (б), разгрузки (в) мельницы и холостого хода мельницы (г – без подачи материала)

Из табл. Ж1 видно, что значение параметра Херста возрастает с увеличением нагрузки струйной мельницы и убывает с уменьшением нагрузки мельницы – во время работы, а также в начале разгрузки параметр максимальный, а когда мельница пустая - минимальный.

Таким образом, определение показателя Херста сигналов акустического мониторинга процесса измельчения позволяет прогнозировать их динамику, и, соответственно, прогнозировать процесс измельчения материала в струйной мельнице.

Исследования показали, что метод нормированного размаха является более чувствительным для оценки выходного сигнала струйной мельницы:

когда мельница без материала, а также когда начинает загружаться, или заканчивает разгружаться параметр Херста $H < 0,5$.

Таблица Ж1

Значения показателя Херста, полученные различными методами

Состояние мельницы	Метод нормированного размаха *(Hrs)*	Метод изменения дисперсий *(Hd)*
Загрузка 1	0,37	0,536
Загрузка 2	0,425	0,562
Загрузка 3	0,538	0,579
Загрузка 4	0.594	0,598
Загрузка 5	0,614	0,638
Рабочий режим 1	0,511	0,566
Рабочий режим 2	0,663	0,642
Рабочий режим 3	0,516	0,597
Рабочий режим 4	0,593	0,619
Рабочий режим 5	0,64	0,612
Разгрузка 1	0,691	0,678
Разгрузка 2	0,509	0,665
Разгрузка 3	0,488	0,592
Разгрузка 4	0,463	0,581
Разгрузка 5	0,439	0,56
Пустая мельница 1	0,467	0,534
Пустая мельница 2	0,497	0,567
Пустая мельница 3	0,33	0,54

Это означает, что в данном случае временной ряд антиперсистентный, то есть ряд, для которого более вероятна смена предыдущего направления. И нагрузка на мельницу в таком случае минимальная. В момент, когда мельница наиболее заполнена – $H > 0,5$, что означает, что в данном случае сигнал

персистентный, то есть ему присуща трендовость (направленность), и нагрузка на мельницу в данный момент максимальная. Чем ближе величина *H* к 0,5, тем более зашумлен ряд и тем менее выражен его тренд.

Метод изменения дисперсии, не смотря на то, что параметры Херста также увеличиваются с возрастанием нагрузки мельницы и убывают с уменьшением нагрузки, является менее чувствительным, так как он не показывает, когда сигнал из антиперсистентного преобразуется в персистентный.

Предложенный подход для контроля уровня загрузки струйной мельницы с использованием вычисления показателя Херста акустического сигнала может обеспечить управление показателями измельчения.

Приложение З

ЗАТВЕРДЖУЮ
проректор з науково-педагогічної роботи
ДВНЗ «Національний гірничий університет»
кандидат геолого-мінералогічних наук,
Хоменко Ю.Т.
10 червня 2015р.

Акт

про впровадження в навчальний процес
результатів дисертаційної роботи
Прядко Наталії Сергіївни
«Розвиток теорії тонкого подрібнення корисних копалин»

«10» 06 2015р. м. Дніпропетровськ

Укладено комісією:
декан факультету інформаційних технологій, д.т.н. Алексєєв М.О.,
заступник завідувача кафедри збагачення корисних копалин, к.т.н. Левченко К.А.

Починаючи з вересня 2012р. на кафедрі програмне забезпечення комп'ютерних систем та на кафедрі збагачення корисних копалин при підготовці дипломних проектів спеціалістів та дипломних робіт магістрів спеціальностей: "Інформаційні управляючі системи та технології", "Програмне забезпечення систем", "Програмна інженерія", «Збагачення корисних копалин» використовуються такі результати дисертаційної роботи:

1) база даних акустичних сигналів зони подрібнення;
2) методика обробки сигналів зони подрібнення та класифікації струминного млина;
3) способи оптимізації тонкого подрібнення;
4) імітаційні моделі тонкого подрібнення в замкнутих системах;
5) методика контролю якості струминного подрібнення;
6) енергетичні особливості тонкого подрібнення.

Впровадження дисертаційної роботи Прядко Н.С. «Розвиток теорії тонкого подрібнення корисних копалин» дозволило підвищити науково-методичний рівень дослідних робіт студентів, надати розрахунковим роботам більше практичного змісту. підготувати біля десяти дипломних робіт за результатами досліджень.

Декан факультету інформаційних технологій, д.т.н. М.О. Алексєєв

Заст. зав.кафедри збагачення корисних копалин, к.т.н. К.А. Левченко

Приложение К

МЕТОДИКИ, АКТЫ ПРОМЫШЛЕННЫХ ИСПЫТАНИЙ И ВНЕДРЕНИЯ НАУЧНО-ТЕХНИЧЕСКОЙ РАЗРАБОТКИ

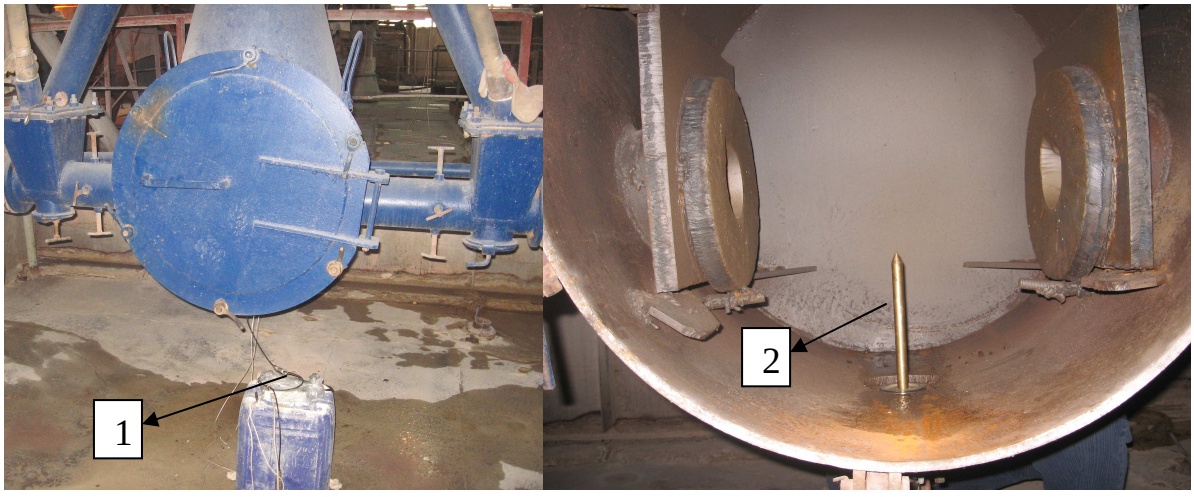
СОГЛАСОВАНО:
Главный инженер ВГМК
_____А.М Лазников
Главный обогатитель ВГМК
_____В.П. Краснопер
_____2011г.

УТВЕРЖДАЮ:
Первый проректор НГУ,
зав. кафедрой ОПИ,
профессор, д.т.н.
_____П.И. Пилов
_____2011г.

АКТ
ПРОМЫШЛЕННОГО ОПРОБОВАНИЯ ОПТИМИЗАЦИИ РАБОТЫ СТРУЙНОЙ ИЗМЕЛЬЧИТЕЛЬНОЙ УСТАНОВКИ НА ОСНОВЕ АКУСТИЧЕСКОГО МОНИТОРИНГА.

Эксперименты проводились согласно разработанной методике (см. Приложение 1). Целью промышленного опробования было подтверждение зависимостей между технологическими и акустическими параметрами зоны помола при измельчении цирконового концентрата. Исходный материал – менее 0,16мм.

Акустическую активность измеряли с помощью широкополосных пьезокерамических датчиков, соединенных с латунным волноводом, установленным внутри помольной камеры и на выходе из классификатора. Для записи и обработки сигналов использовался аналого-цифровой преобразователь (АЦП) и компьютер (ноутбук). С учетом экспериментов, проведенных в условиях ВГМК в 2011г., форма волновода усовершенствована. На рис. К1 показана установка акустической аппаратуры на промышленной струйной мельнице.



а) б)

Рисунок 1. –Установка АЦП, датчика (а) и волновода (б) в помольной камере: 1- АЦП, 2 – волновод.

Исследовались режимы измельчения и акустической активности мельницы в начале загрузки струй материалом, в рабочем режиме, при разгрузке струй и перегрузке мельницы. Изучалось влияние давления энергоносителя

(P = 0,52; 0,4 и 0,3 МПа) и частоты вращения ротора классификатора (n = 107, 177 мин$^{-1)}$ на акустические параметры зон помольной камеры. Загрузка исходным материалом проводилась непрерывно, однако ее уровень регулировался в ходе эксперимента.

Условия и результаты испытаний приведены в таблице К1.

Таблица К1

Условия и результаты опробования акустического мониторинга на мельнице №7 Вольногорского ГМК

t, мин	n, мин$^{-1}$	P, МПа	Sуд, см^2/г	Q, кг/ч	$R_{60,}$ $R_{40,\%}$	К загр	A_{max}, мВ	$A_{ср}$, мВ	$\dot{N}$ *10^5, с$^{-1}$	Режимы
0	86	0,6	708		R_{60}		40	35	1,3	Подготов
28	84	0,6		960		1	200	65	0,67	Перегруз
45	84	0,6	1515		2,3	1	190	75	0,96	Рабочий
69	84	0,6		1020		0,5	165	90	0,87	Рабочий
85	84	0,6		1100	2,0	0,5	185	100	0,9	Рабочий
		0,6	1430		3,2		155	80	0,85	Недогруз
95	84	0,5- 0,4- 0,3				1	130 50 10	70 20 5	0,94 0,92 0,9	Изменение Р,
102	180	0,61	2376	600	1,0	1	180	105	0,92	Рабочий
130	200	0,61			0,8	0,5	170	97	0,82	недогруз
131		0,5- 0,4- 0.3				1	120 40 10	65 10 5	0,9 0,85 0,73	Изменение Р

Обработка результатов акустического мониторинга работы струйной мельницы включала установление следующих параметров и зависимостей:

- кинетика акустической активности $\dot{N}$ и амплитуд средних и максимальных значений (A_{max} , $A_{ср}$) акустических сигналов в рабочем режиме измельчения;

- амплитудное распределение акустических сигналов $\dot{N}$ на стадии разгрузки струй в зависимости от режима классификации (см. рис. К2);

- амплитудное распределение акустических сигналов $\dot{N}$ в рабочем режиме при различных значениях давления энергоносителя (см. рис. К3);

- кинетика амплитуды сигналов от режима измельчения: загрузка, рабочий режим и разгрузка (см. рис. К4)

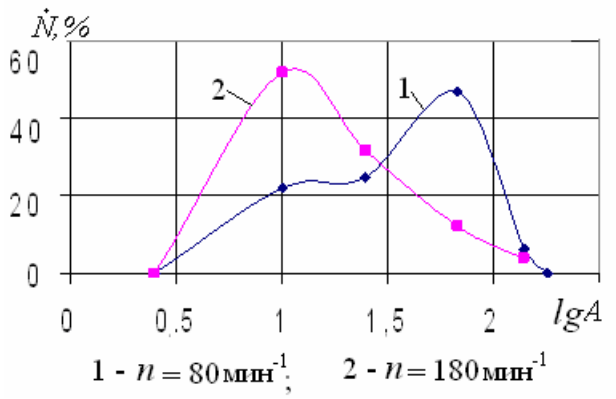


Рисунок К2. – Амплитудное распределение АС в рабочем режиме зоны помола в зависимости от режима классификации.

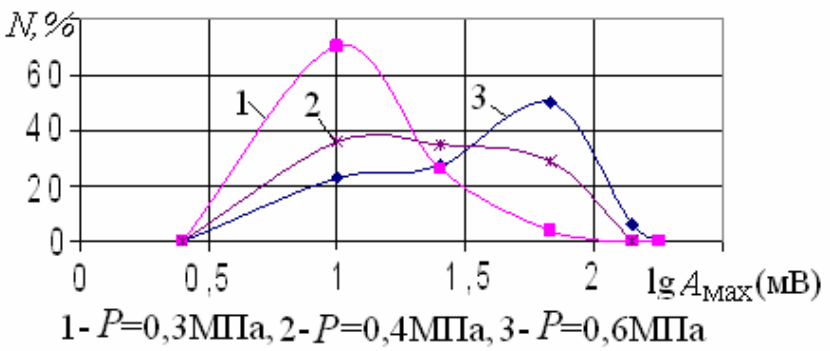


Рисунок К3. – Амплитудное распределение АС зоны помола в условиях изменения давления P энергоносителя (n = 80мин^{1}).

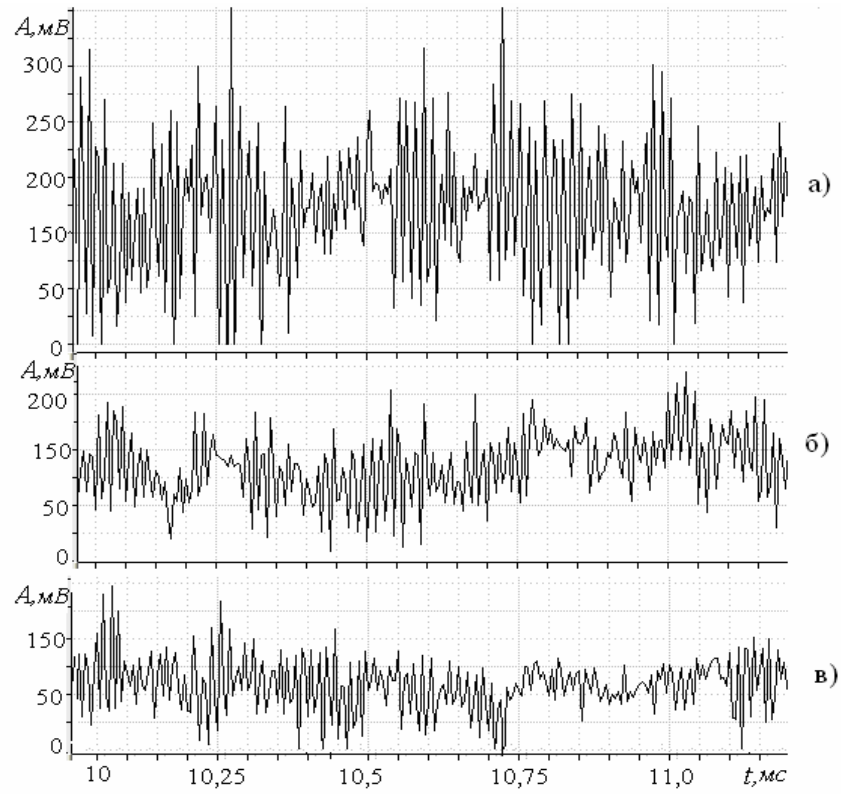


Рисунок К4. – Кинетика амплитуд акустических сигналов зоны помола при загрузке струй а), в рабочем режиме б) и при разгрузке струй в).

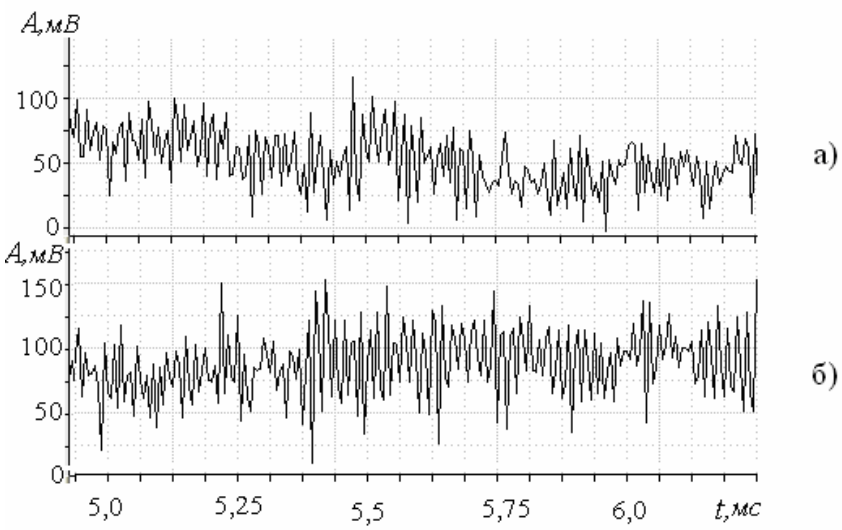


Рисунок К5. – Влияние частоты вращения ротора классификатора на величину амплитуды акустических сигналов зоны помола в рабочем режиме:
а – n = 180 с$^{-1}$, $A_{ср} \approx$ 45 мВ; б – n =80 с$^{-1}$ $A_{ср} \approx$ 65 мВ.

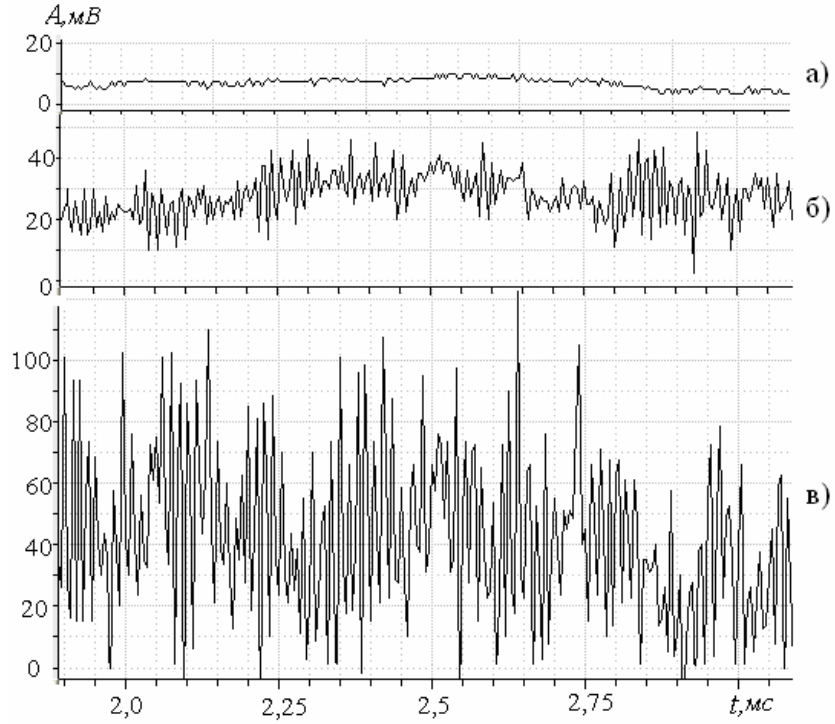


Рисунок К6. – Влияние давления P энергоносителя на величину амплитуд акустических сигналов зоны помола промышленной мельницы.
а – P = 0,3МПа , $A_{ср} \approx$ 5мВ;
б – P = 0,4 МПа, $A_{ср} \approx$ 20мВ;
в – P = 0,6 МПа, $A_{ср} \approx$ 70мВ.

Исследования показали, что акустические параметры ($\dot{N}$, $A_{ср}$, $A_{мах}$) зоны помола изменяются под влиянием степени загрузки струй материалом (перегрузка, недогрузка, рабочий режим), частоты вращения ротора классификатора и давления энергоносителя.

Экспериментально установлено, что акустические параметры сигналов потока измельченного продукта в воздуховоде после классификатора изменяются в зависимости от режима классификации, т.е. величины частиц полученного измельченного продукта (см. рис. К7) .

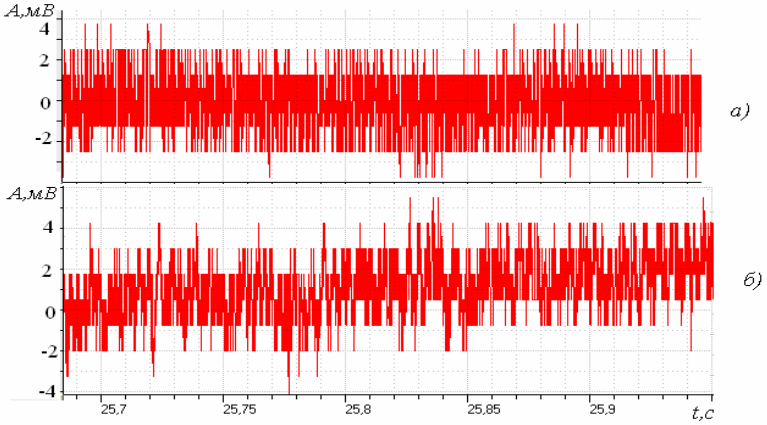


Рисунок К 7. – Влияние частоты вращения ротора классификатора на величину амплитуды акустических сигналов в зоне выхода классификатора (рабочий режим):

а – n = 200 мин-1, Аср ≈ 2,5 мВ, Амах = 3,8 мВ;

б – n = 84 мин-1 , Аср ≈ 3 мВ, Амах = 4,2-5,5 мВ.

Причем, исследования показали, что по величине амплитуды возможно определить изменение гранулометрического состава продукта в ходе измельчения и, таким образом, добиться более качественного измельченного продукта. На рис. К8 показано изменение амплитуды акустических сигналов на выходе из классификатора при фиксировании частиц большего размера. При промышленном измельчении в этом случае контролировали качество продукта изменением числа оборотов классификатора (с 180 до 200мин$^{-1}$).

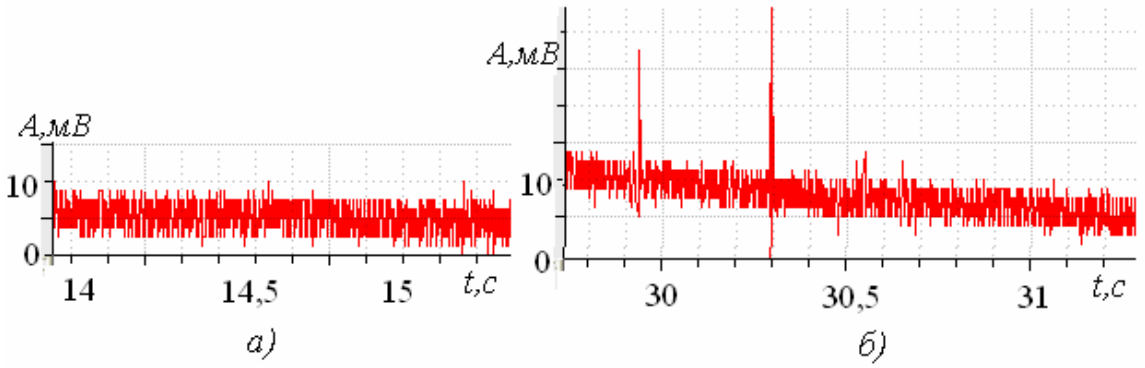


Рисунок К8 – Изменение амплитуды акустических сигналов на выходе из классификатора (n=180мин$^{-1}$): а) $A_{мах}$ = 7 мВ, $A_{ср}$ = 3 мВ; б) $A_{мах}$ = 17 - 30мВ, $A_{ср}$ = 3мВ

Для измерения количества и качества акустических сигналов волновод был установлен в центре помольной камеры. В ходе опробования было измельчено порядка двух тонн материала в течение 2,1 часа. Из фотографии рис. К9 виден значительный износ волновода (≈2см). Это указывает на необходимость некоторого удаления волновода из зоны его интенсивного износа

и проверки влияния нового местоположения на акустические характеристики зоны помола.

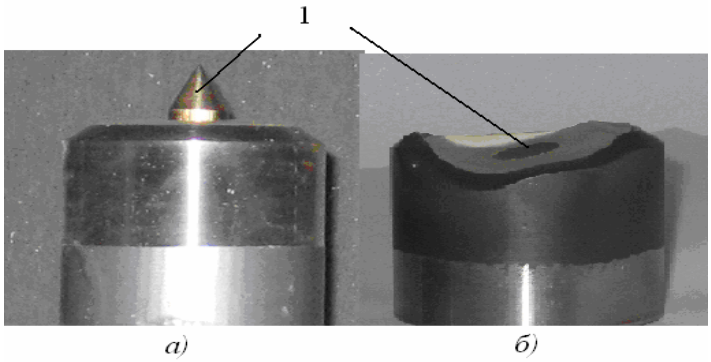


Рисунок 9. – Износ волновода 1 в результате взаимодействия с частицами: а) – волновод до измельчения, б) - после измельчения 2 т материала.

Считаем целесообразным проведение исследования по установлению рационального места волновода в помольной камере. Кроме того, проведенное опробование показало, что необходимо увеличить число измерений текущей производительности Q для установления статистических показателей и дальнейшего анализа связи Q и акустических параметров.

Таким образом, исследования показали перспективность применения акустического мониторинга для задачи оптимизации процесса измельчения. Направлениями дальнейших испытаний могут быть:

- создание информационной системы контроля акустических параметров промышленной струйной мельницы;
- обоснование критериев поиска и поддержания оптимальной работы мельницы,
- разработка алгоритма и программы обработки данных акустического мониторинга с целью снижения удельных энергозатрат путем оптимизации режима измельчения.

От ВГМК	От НГУ, ИТМ НАНУ и НКАУ
Старший мастер цеха обогащения	Проф. кафедры ОПИ, д.т.н
Ю.В. Новодон	Л.Ж. Горобец
Начальник участка	Ст. научн. сотр. отд. 6, к.т. н.
С.И. Мирчун	Н.С. Прядко

М Е Т О Д И К А

ПРОВЕДЕНИЯ АКУСТИЧЕСКОГО МОНИТОРИНГА РАБОТЫ СТРУЙНОЙ МЕЛЬНИЦЫ В УСЛОВИЯХ ВОЛЬНОГОРСКОГО ГОРНО-МЕТАЛЛУРГИЧЕСКОГО КОМБИНАТА

Целью экспериментов является промышленное опробование принципов акустического мониторинга и оптимизации работы мельницы при измельчении цирконового концентрата.

На рис. 1 и рис. 2 показана схема и блок-схема акустического мониторинга струйной измельчительной установки. Акустические параметры рабочей зоны измельчения измеряли с помощью широкополосных пьезокерамических датчиков. Фиксирование, запись и обработка сигналов осуществлялась посредством аналого-цифрового преобразователя (АЦП) типа E14-440, который соединялся с персональным компьютером (ПК) проводом с USB-выходом.

Методика проведения акустического мониторинга зоны измельчения включает следующие этапы.

В различных режимах измельчения (загрузке струй материалом, рабочем режиме и разгрузке струй) измеряется общее число акустических сигналов (АС) различных амплитуд. Длительность опыта с отбором измельченной пробы за 10мин и определением производительности должна составить порядка 40мин. Устанавливаются значения амплитуд и числа акустических сигналов, характеризующих режим истечения струй в различных режимах загрузки материала.

Обработка опытных данных акустического мониторинга работы струйной мельницы включает установление следующих параметров и зависимостей:

- кинетику акустической активности при измельчении;
- число АС с максимальными значениями амплитуд при высокой акустической активности;
- долю счета акустических сигналов с малыми значениями амплитуд на стадии разгрузки струй;
- построение амплитудных распределений АС по величине для установления отклонений процесса от оптимального рабочего режима (путем перегрузки или разгрузки струй).

В процессе измельчения производится отбор исходных и измельченных проб, согласно запланированным испытаниям, последовательность которых показано в таблице 1.

Таблица 1

Условия и порядок проведения мониторинга работы промышленной струнной мельницы в условиях ВГМК

№ п/п	Содержание испытаний	№ пробы	Примечание
1	Отбор исходной пробы Акустические параметры холостого хода Режим измельчения: P =0,5=0,6 МПа* Режим классификатора: n = 80-200 мин$^{-1}$ Режим загрузки: условия производства	1	загрузка
2	Мониторинг кинетики измельчения в рабочем режиме при постоянных параметрах	2	Рабочий режим
3	Перекрыть подачу исходного материала Мониторинг разгрузки струй		Разгрузка
4	Открыть подачу материала. Мониторинг загрузки струй с выходом на рабочий режим		Рабочий режим
5	Увеличить n = с 80 до 120 мин$^{-1}$ Мониторинг перегрузки струй		Перегрузка
6	Установить $n_{норм}$ = 80 мин$^{-1}$ Мониторинг кинетики рабочего режима		Рабочий режим
7	Уменьшить давление с 0,6 до 0,3 МПа Мониторинг перегрузки струй		Перегрузка
8	Возврат к $P_{норм}$ =0,5-0,6 МПа Мониторинг перехода к рабочему режиму		Рабочий режим
9	Подведение итогов экспериментов. Определение производительности, анализ проб, обработка акустической информации и технологических данных. Обобщение выводов и разработка рекомендаций. Составление акта испытаний		Анализ результатов Акт испытаний

* - численные значения параметров в процессе испытаний могут корректироваться

Испытания предусматривают определение производительности при различных изменениях режимных параметрах (частоты вращения n классификатора и давления энергоносителя). При этом пробы необходимо оценить на гранулометрический состав и величину удельной поверхности на приборах ВГМК и НГУ (MALVERN и Т-3).

На рис. 3 показана установка акустической аппаратуры на промышленной струйной мельнице. В днище или боковую крышку устанавливается акустическое принимающее устройство 1, включающее волновод и датчик. Это устройство соединено USB-кабелем с АЦП 2 и компьютером 3 (ноутбуком).

Акустические сигналы записываются и сохраняются в режиме on-line. Акустический мониторинг проводится по основным акустическим параметрам: амплитуде и активности сигналов. В ходе дальнейшего анализа по специальным программам вычисляются необходимые технологические параметры и критерии: максимальная амплитуда A_{max} *(B),* активность акустических сигналов $\dot{N}$ (имп/с), производительность мельницы Q (кг/ч)), критерий оптимальной работы мельницы *Ка.*

На рис. 4 показано схематическое поведение исследуемых параметров в процессе измельчения при загрузке порций *m* исходного материала. По оси абсцисс обозначено текущее время в различных состояниях струй: загрузка, оптимальный рабочий режим, перегрузка, рабочий режим (р/р), разгрузка.

Результаты экспериментального промышленного опробования (после обработки акустической информации и оценки дисперсности измельченных продуктов) оформляются в виде акта промышленных испытаний с акустическим мониторингом работы струйной измельчительной установки.

СОГЛАСОВАНО представителями НГУ, ИТМ НАНУ и НКАУ и ВГМК

От ВГМК

Главный обогатитель И.В. Самофал

инженер С.И. Мирчун

От НГУ, ИТМ НАНУ и НКАУ

профессор, д.т.н Л.Ж. Горобец

с.н.с., к.т.н Н.С. Прядко

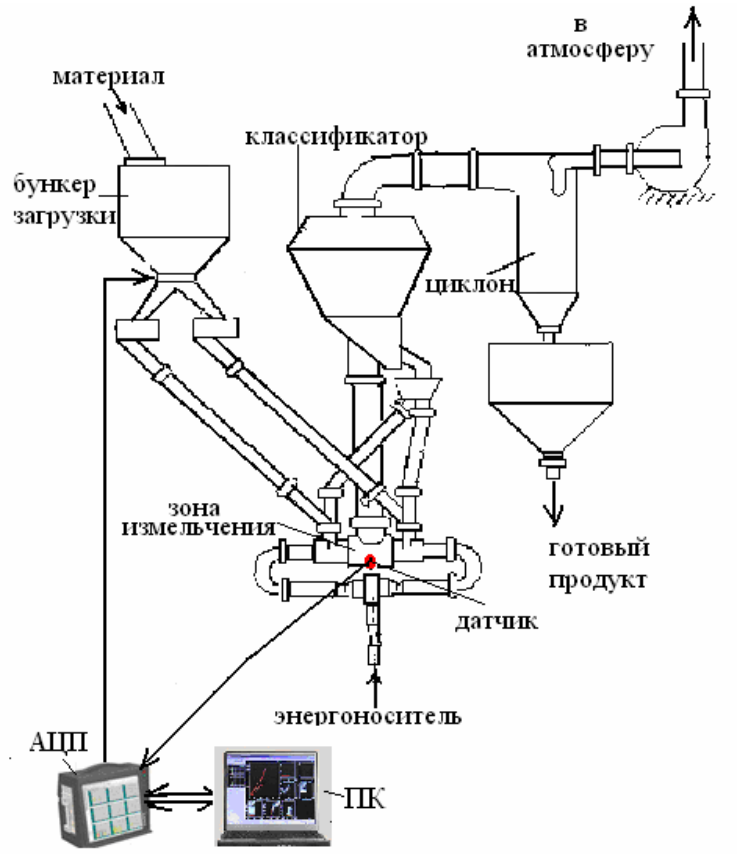


Рисунок 1. – Схема акустического мониторинга струйной измельчительной установки.

Рисунок 2. – Блок - схема акустического мониторинга зоны помола.

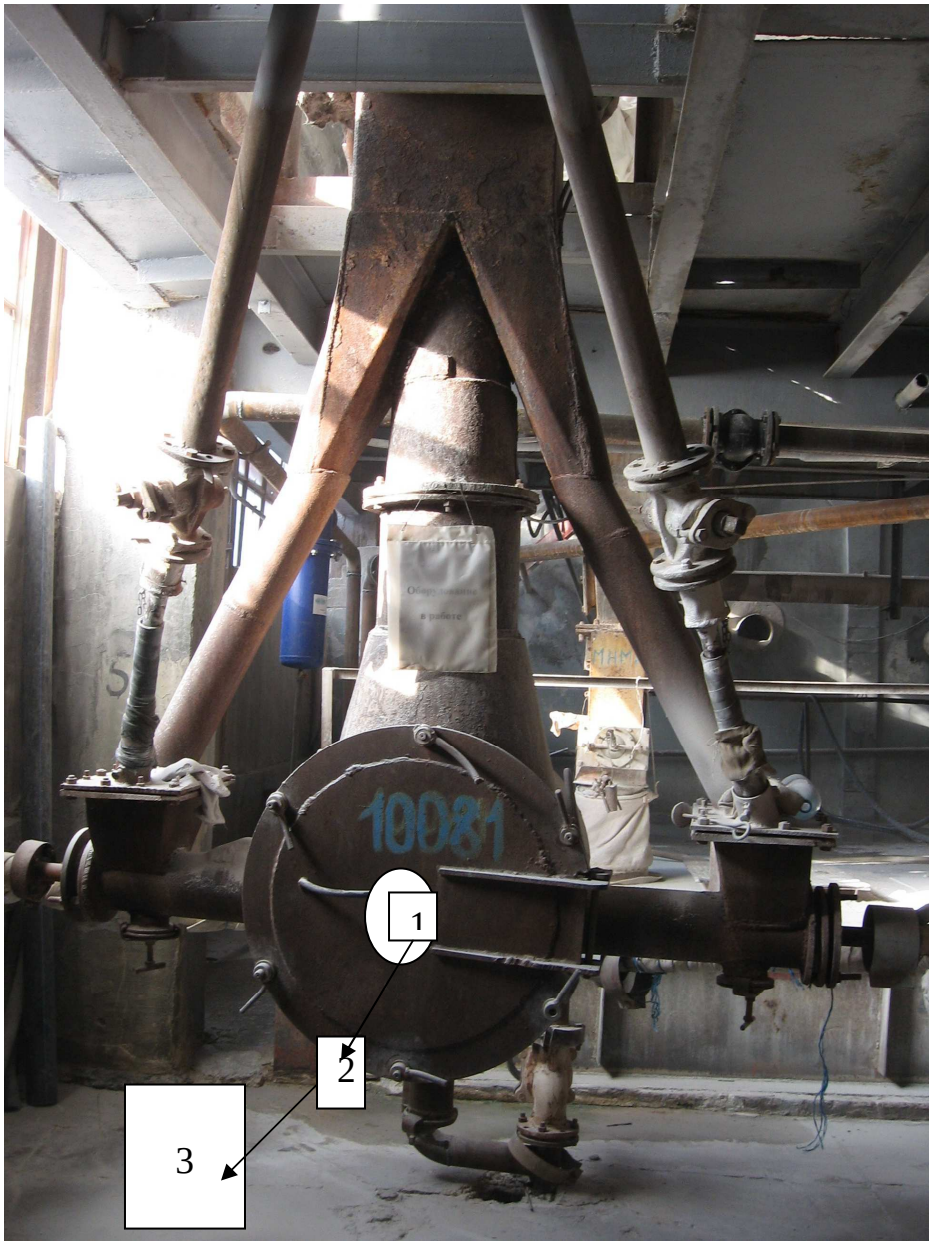


Рисунок 3 – Установка акустической аппаратуры на промышленной струйной мельнице

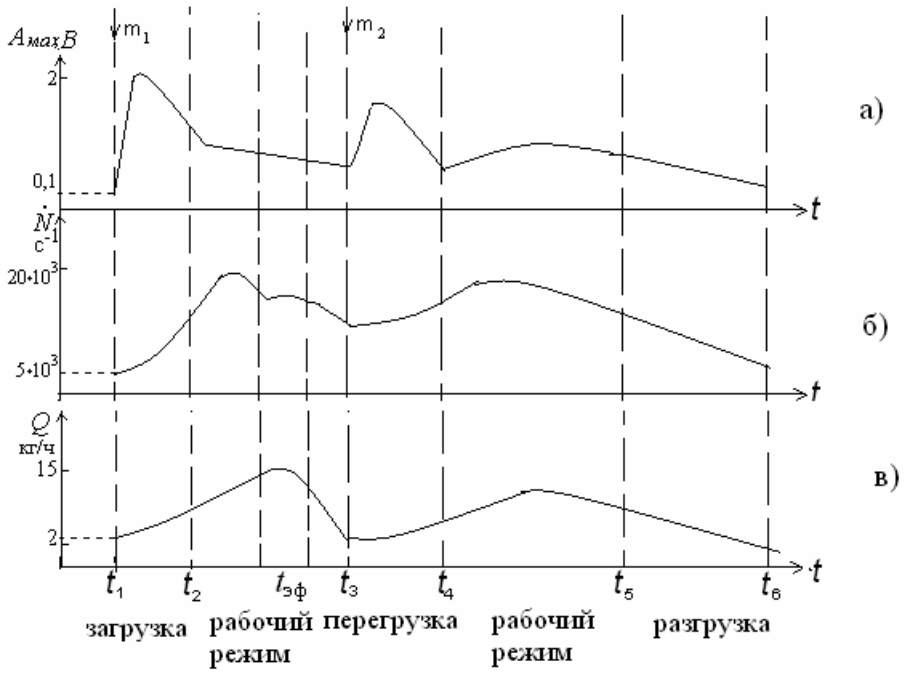


Рисунок 4 – Схематическое изображение информативных акустических и технологических параметров процесса струйного измельчения: максимальной амплитуды A_{max} *(B)*, активности акустических сигналов $\dot{N}$ имт/с), производительности мельницы *Q* (кг/ч).

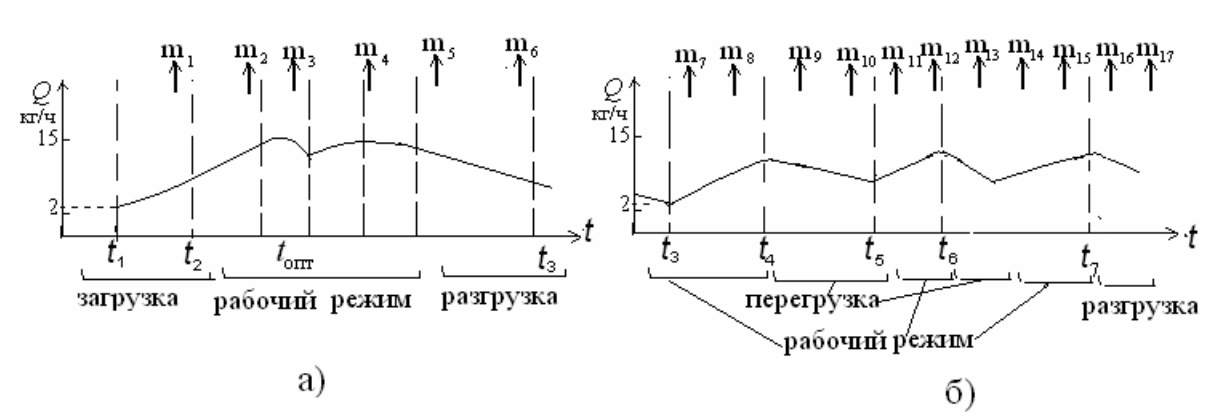


Рисунок 5 – Схематическое изображение изменения производительности мельницы на различных этапах ее работы в соответствии с табл. 1:
а) загрузка, рабочий режим измельчения, разгрузка (режим 1 - 3)
б) рабочий режим, перегрузка, рабочий режим (режим 4 -8)

«Вольногорский горно-металлургический комбинат» ОБОГАТИТЕЛЬНАЯ ФАБРИКА

УТВЕРЖДАЮ:
Главный обогатитель ВГМК
_____ И.В. Самохвал
15. 11. 2012г.

Приложение № 1
к методике проведения акустического мониторинга на ВГМК

ПОРЯДОК ПРОВЕДЕНИЯ АКУСТИЧЕСКИХ ИЗМЕРЕНИЙ ХАРАКТЕРИСТИК ПРОТИВОТОЧНОЙ СТРУЙНОЙ МЕЛЬНИЦЫ

Эксперименты проводятся согласно методике проведения промышленных экспериментов по определению акустических характеристик противоточной струйной мельницы. Данная методика утверждена 22.10.2012г. первым проректором НГУ П.И. Пиловым после согласования с главным инженером ВГМК В.П. Краснопером и главным обогатителем ВМК И.В. Самохвалом.

Эксперименты проводятся с целью промышленного опробования принципов оптимизации работы мельницы и контроля качества получаемого продукта. Исследуется режимы загрузки мельницы, а также давление энергоносителя (P = 0,6; 0,5; и 0,3 МПа) и частота вращения ротора классификатора (n = 107, 177 мин$^{-1}$) и акустические параметры (амплитуды и акустическая активность) в помольной камере и на выходе классификатора.

Условия планируемого эксперимента приведены в таблице 1.

Таблица 1

№ п/п	N пробы	Δt, мин	$M_{проб}$, кг	Q, кг/ч	$A_{камера}$/ $A_{классиф}$,В	$\dot{N}_{кам}$ / $\dot{N}_{классиф}$	режим	Примечание
1	1	30	0,1	Q1			рабочий	
2	2	30	0,1	Q2			разгрузка	Нет подачи материала
3	3	30	0,1	Q2			рабочий	
4	4	20	0,1	Q4			перегрузка	
5	5	10	0,1	Q5			перегрузка	Изменение давления

В результате проведенных экспериментов проводятся измерения $S_{уд}$, г/см^2 и грансостава исходного концентрата и измельченных продуктов на приборе MALVERN.

По итогам экспериментов составляется отчет и разрабатываются рекомендации по практическому использованию результатов. Информация предоставляется всем участвующим сторонам.

СОГЛАСОВАНО представителями НГУ, ИТМ НАНУ и НКАУ и ВГМК

От ВГМК С.И. Мирчун

От НГУ, ИТМ НАНУ и НКАУ Л.Ж. Горобец
Н.С. Прядко

УТВЕРЖДАЮ
Главный инженер Вольногорского
горно-металлургического комбината
Краснопер В.П.
2014г

Акт
о промышленном внедрении научно-технической разработки.

Мы, нижеподписавшиеся, главный инженер ВГМК Краснопер Валерий Петрович, главный обогатитель ВГМК Самофал Игорь Владимирович, представитель ВУЗ «Национальный горный университет», профессор кафедры ОПИ Горобец Лариса Жановна и старший научный сотрудник ИТМ НАНУ и ГКАУ Прядко Наталья Сергеевна, составили настоящий акт в том, что в соответствии с договоренностью ВУЗ «НГУ» и ИТМ НАНУи ГКАУ проведены промышленные опробования (2010-2014 гг.) акустического мониторинга работы струйной мельницы. Методика приведена в Приложении К.

В ходе измельчения циркона в струйной установке №7 подтверждены связи технологических и акустических параметров процесса и установлены технолого-акустические критерии оценки степени оптимальности режима работы мельницы (Приложение К2).

Внедрение системы оптимизации процесса на основе акустического мониторинга позволит обеспечить повышение производительности мельницы на 50 - 70 % в зависимости от дисперсности измельченного продукта (удельная поверхность в диапазоне $S_{уд}^{=}$ 1540 - 2500 см^2/г).

Удельная экономия электроэнергии на 1 тонну готового продукта (при цене 0,44 грн/квт-ч) от внедрения системы акустической оптимизации может составить, соответственно:

- для продукта менее -63 мкм ($S_{уд}^{=}$ 1540 – 1750 см^2/г) – 32 - 60 грн/т;
- для продукта менее -45 мкм ($S_{уд}^{=}$ 2500 – 2200 см^2/г) – 176 - 200 грн/т;

Ожидаемый годовой экономический эффект может составить величину порядка 185 – 507 тыс грн, соответственно для получаемого измельченного циркона крупностью 63 и 45мкм.

измельченного циркона крупностью 63 и 45мкм.

Главный обогатитель ВГМК Самофал И.В.

профессор кафедры ОПИ
ВУЗ «НГУ» Горобец Л.Ж.
старший научный сотрудник
ИТМ НАНУ и ГКАУ Прядко Н.С.

ПРИЛОЖЕНИЕ К2

Параметры и показатели оптимальной работы струйной мельницы № 7 ВГМК

Показатели оценки акустической ($Э_{изм}$, $Э_{дисп}$) и технологической ($E_{уд}$, E_s) энергоемкости измельчения и диспергирования циркона в различных режимах работы промышленной мельницы производительностью G при получении продукта удельной поверности $S_{уд}$.

G, кг/ч	$S_{уд}$, $см^2/г$	$Э_{изм}$, имп/г	$Э_{дисп}$, имп/$см^2$	$E_{уд}$, квт-ч/т	E_s, кВт-ч/ $м^2$
Оптимальный режим, n= 80 - 107 $мин^{-1}$					
1416	1753	381,36	0,218	201,27	0,00115
1406	1809	409,67	0,226	202,70	0,00112
1138	1861	556,77	0,299	250,44	0,00135
902,4	1911	678,19	0,355	315,82	0,00165
1020	1515	321,18	0,212	279,41	0,0018
Оптимальный режим, n= 170 - 180 $мин^{-1}$					
1094,4	2513	450,67	0,179	260,4167	0,00104
1022,4	2342	669,01	0,286	278,76	0,00119
624	2605	1038,46	0,399	456,73	0,00175
Неоптимальный режим, n= 80 - 107 $мин^{-1}$					
854	1536	843,09	0,55	333,72	0,0022
734,4	1547	882,35	0,57	388,07	0,0025
Неоптимальный режим, n= 170 - 180 $мин^{-1}$					
432	2209	1416,67	0,64	659,72	0,00298
354	2451	1830,51	0,75	805,08	0,00328

Расчеты показателей:

Прирост производительности G:

-63мкм: (1400-734)/1400*100%=47,6%

-45мкм: (1094-354)/1094*100%=67,6%

Экономия электроэнергии:

-63мкм: (334-200)*0,44=60 грн/т и (388-316)*0,44=32 грн/т;

- 45 мкм: (660 – 260)*0,44=176 грн/т.

Годовой экономический эффект (число часов работы в году 15*12*16=2880):

-63мкм - 2880*1,4 * 4032 (32+60)/2 =185 000грн

-45мкм 2880*1,0*176= 507 000 грн.